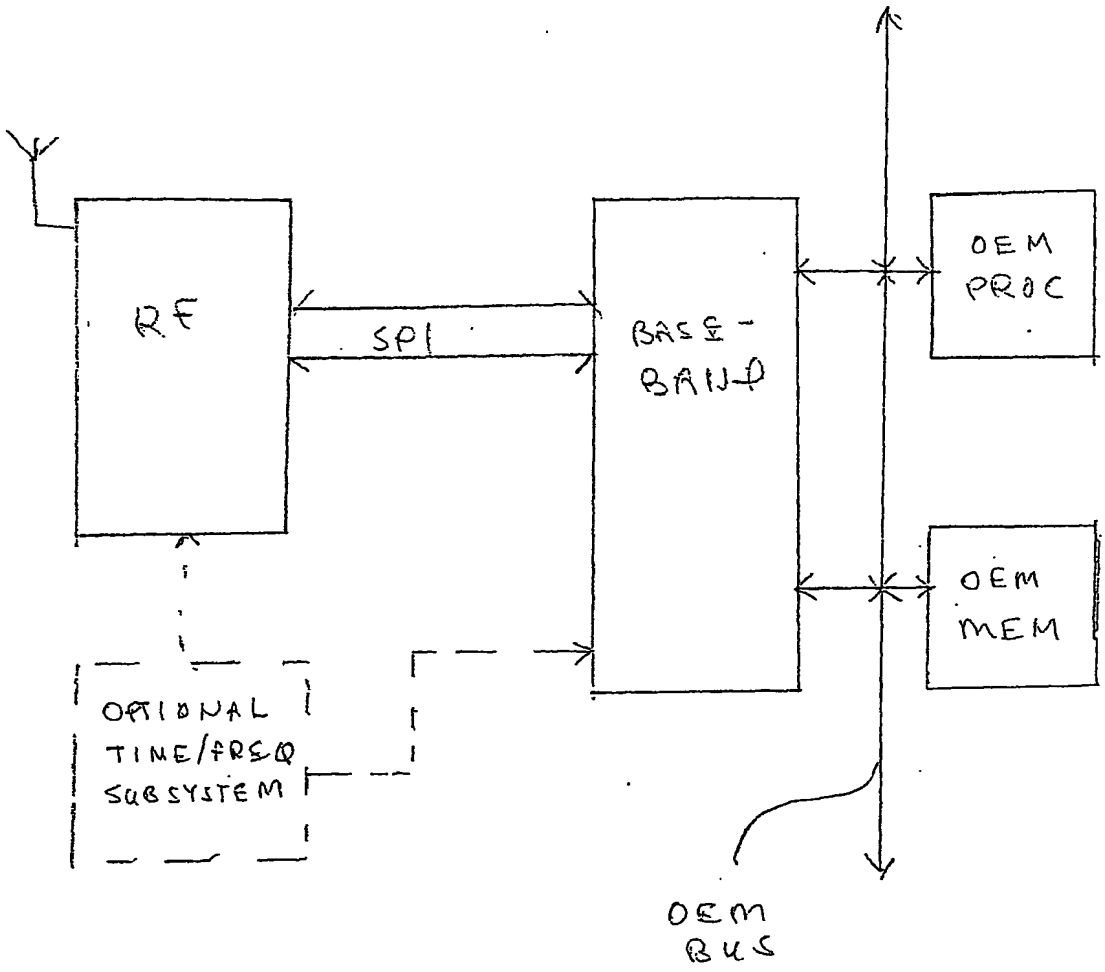

RF
SPI
BASE-BAND
OEM PROC
OEM MEM
OPTIONAL TIME/FREQ SUBSYSTEM
OEM BUS

FIG. 1

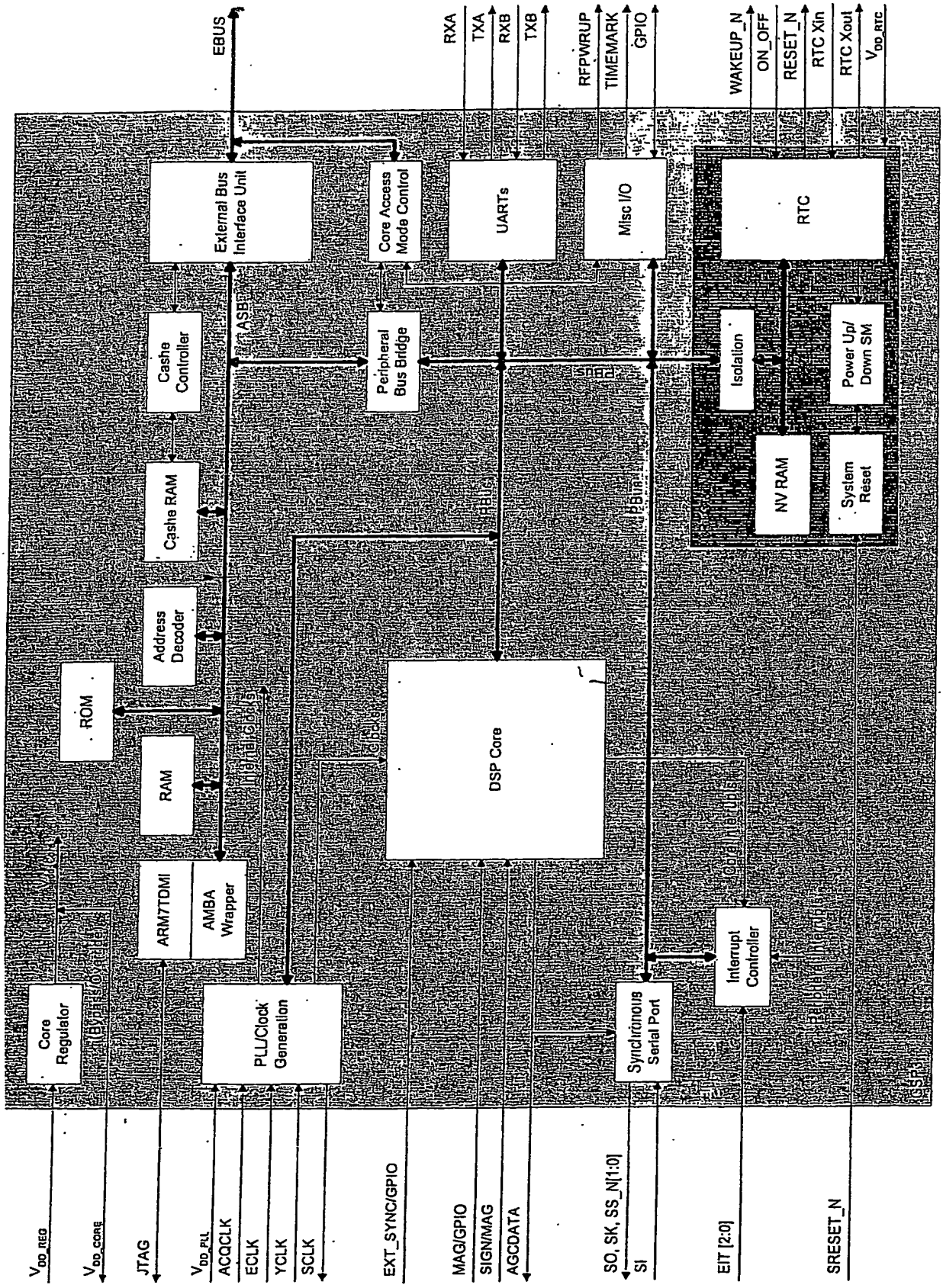


FIG. 1A

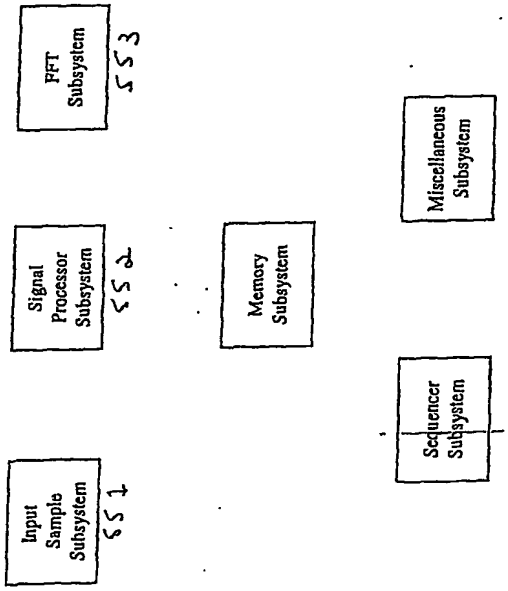
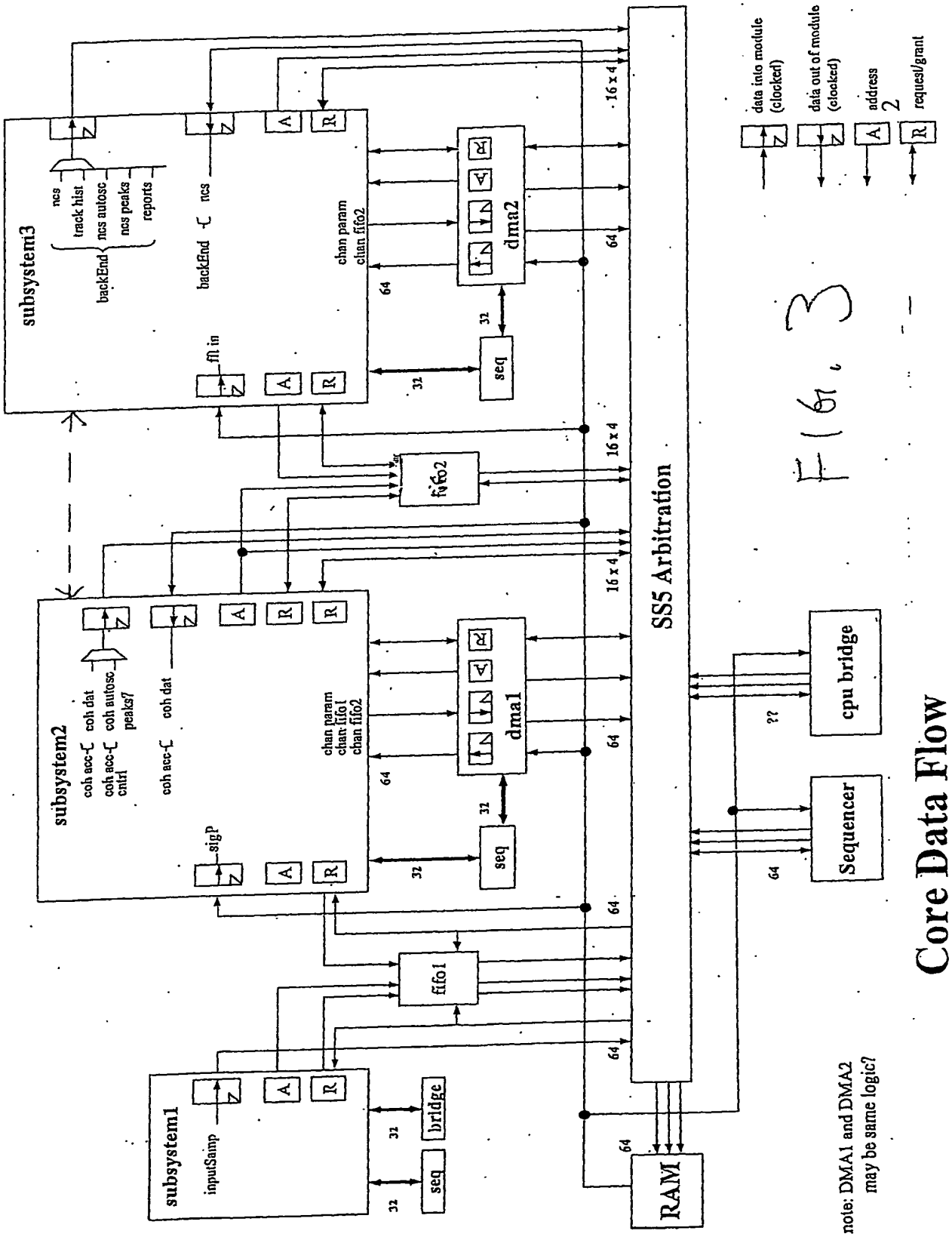

SiRFStar3 Core Major Subsystem
Input Sample Subsystem
Signal Processor Subsystem
FFT Subsystem
Memory Subsystem
Miscellaneous Subsystem
Sequencer Subsystem

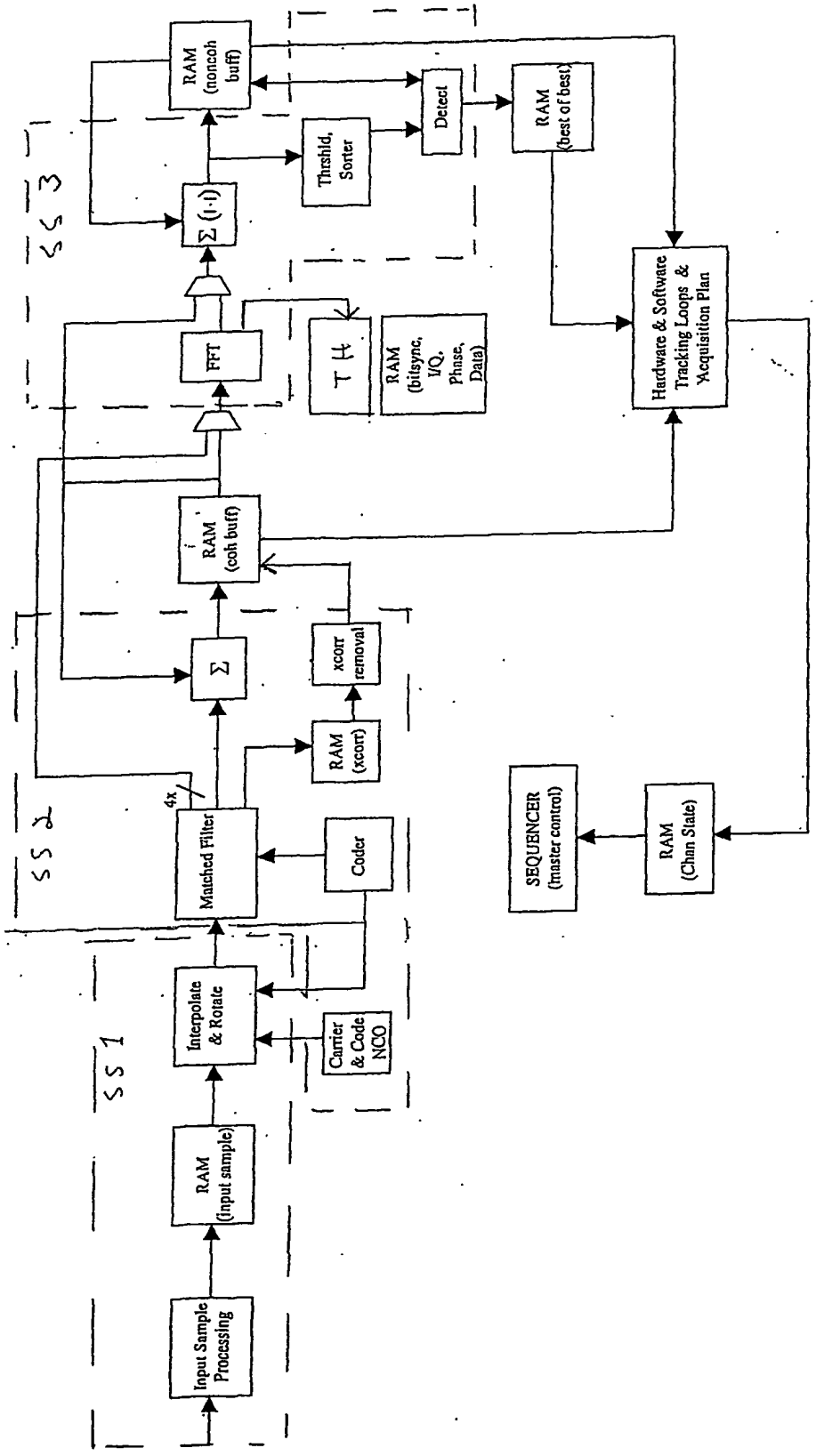
FIG. 2



Core Data Flow

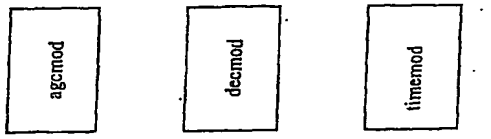
note: DMA1 and DMA2 may be same logic?

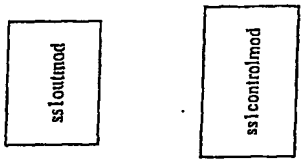
FIG. 3



F16. 4

General Data Flow





input sample subsystem partitioning

FIG. 5

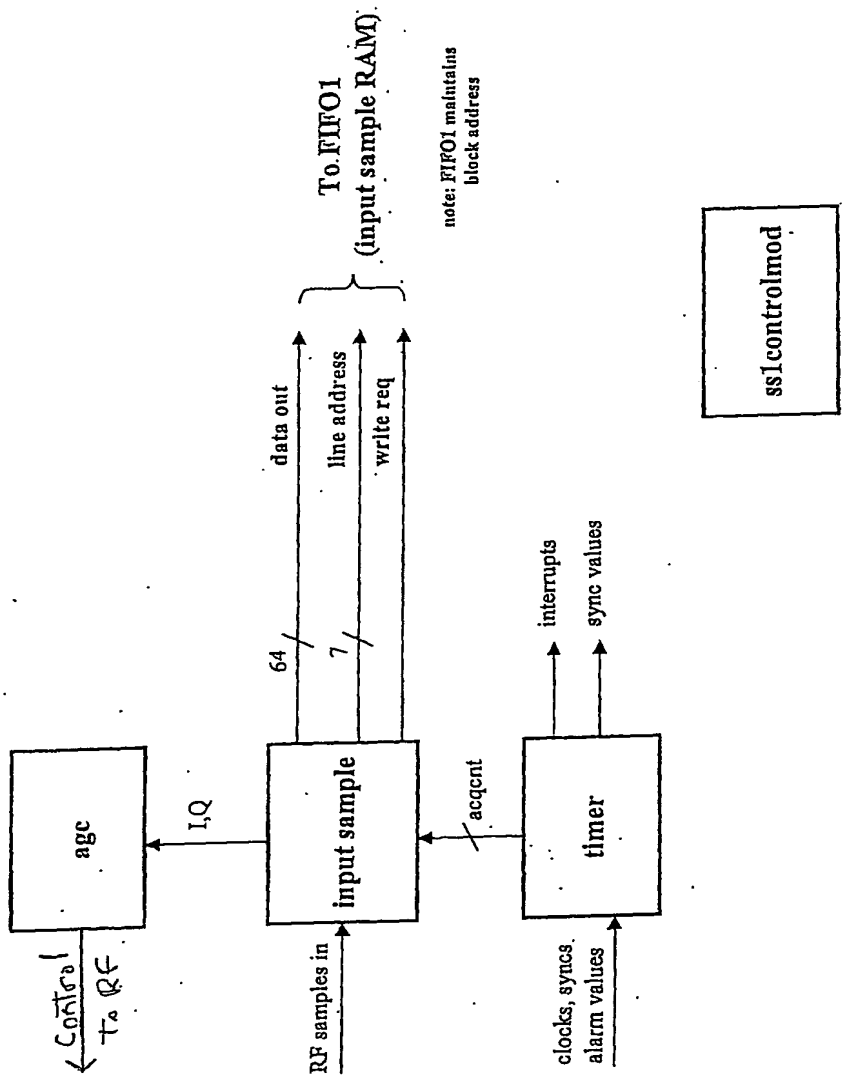


FIG. 6

Subsystem 1 Partitioning

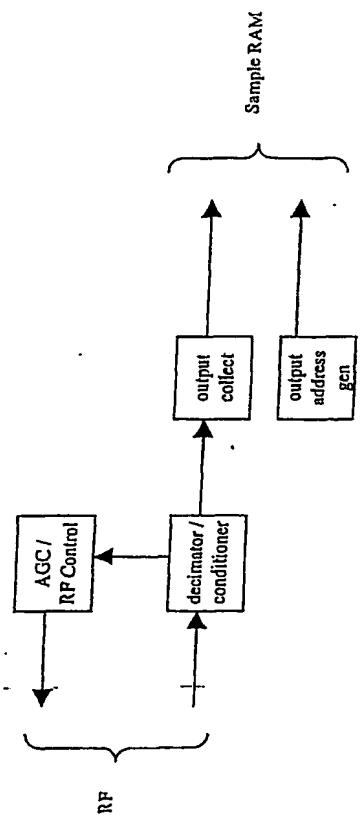


Fig. 7

input sample subsystem flow

signal processor subsystem partitioning
ss2inmod
sigprocmod
codermod
mifmod
cohacmod
ss2outmod
ss2controlmod

FIG. 8

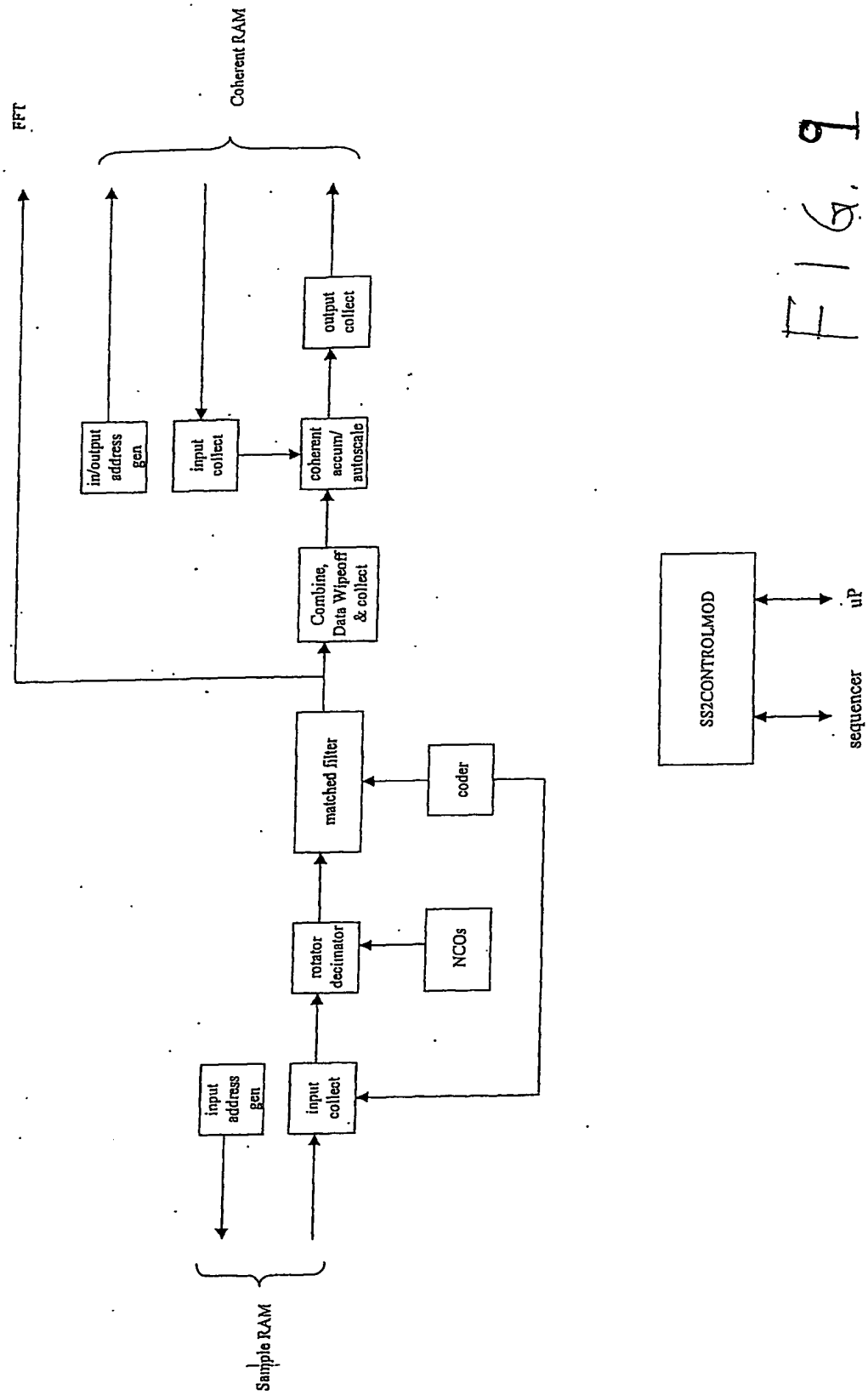
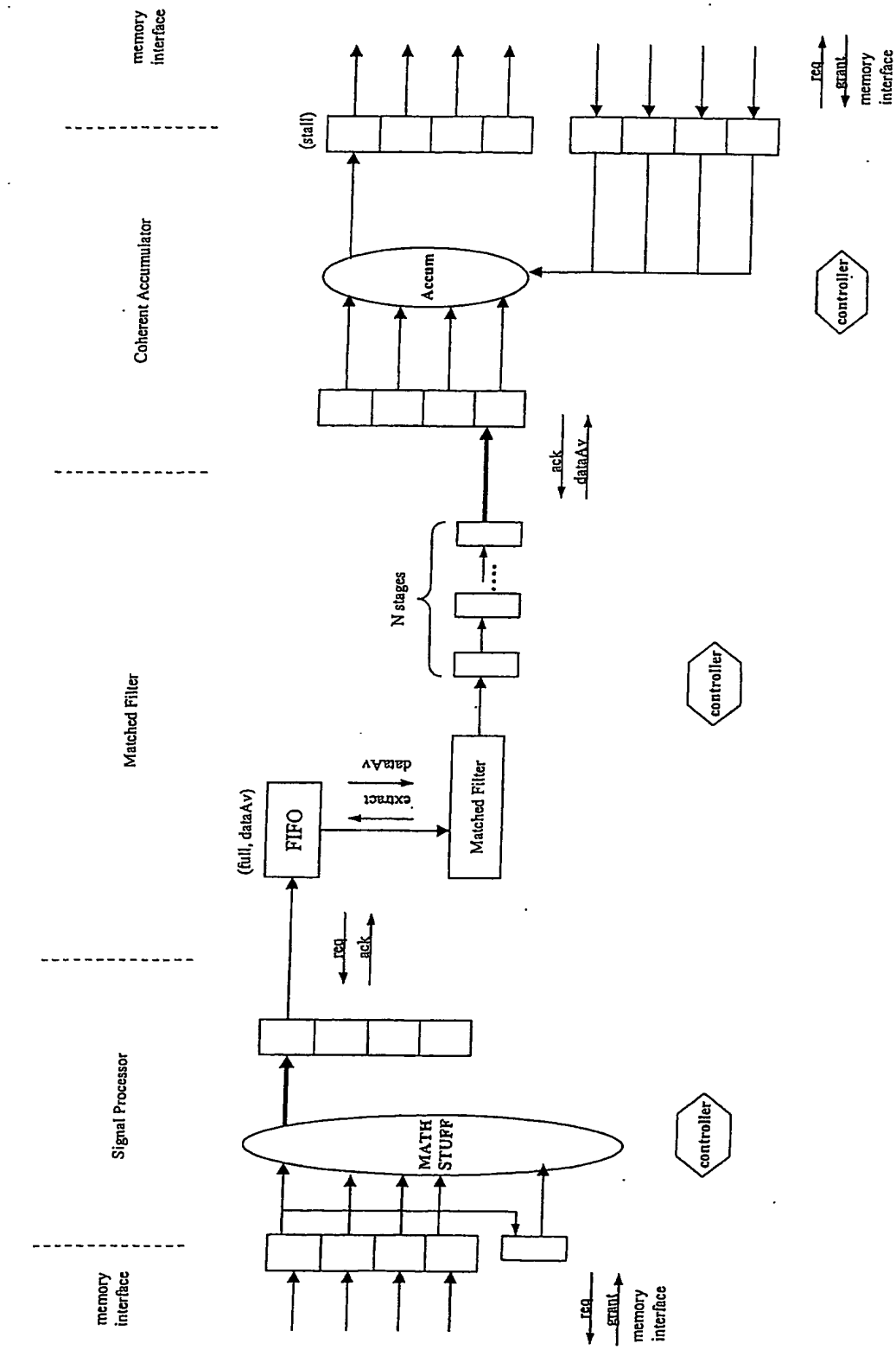
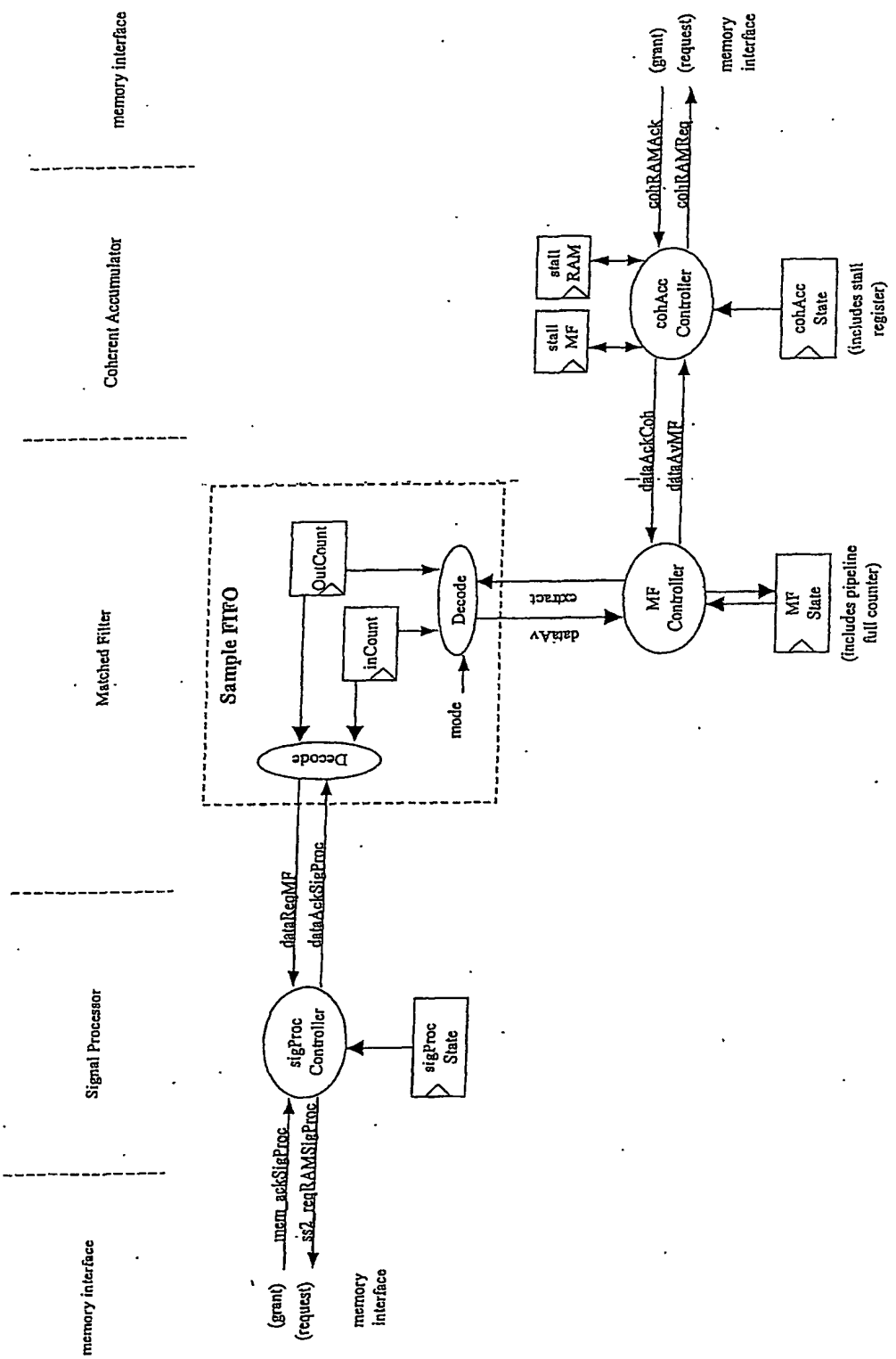


Fig. 9

signal processor subsystem flow

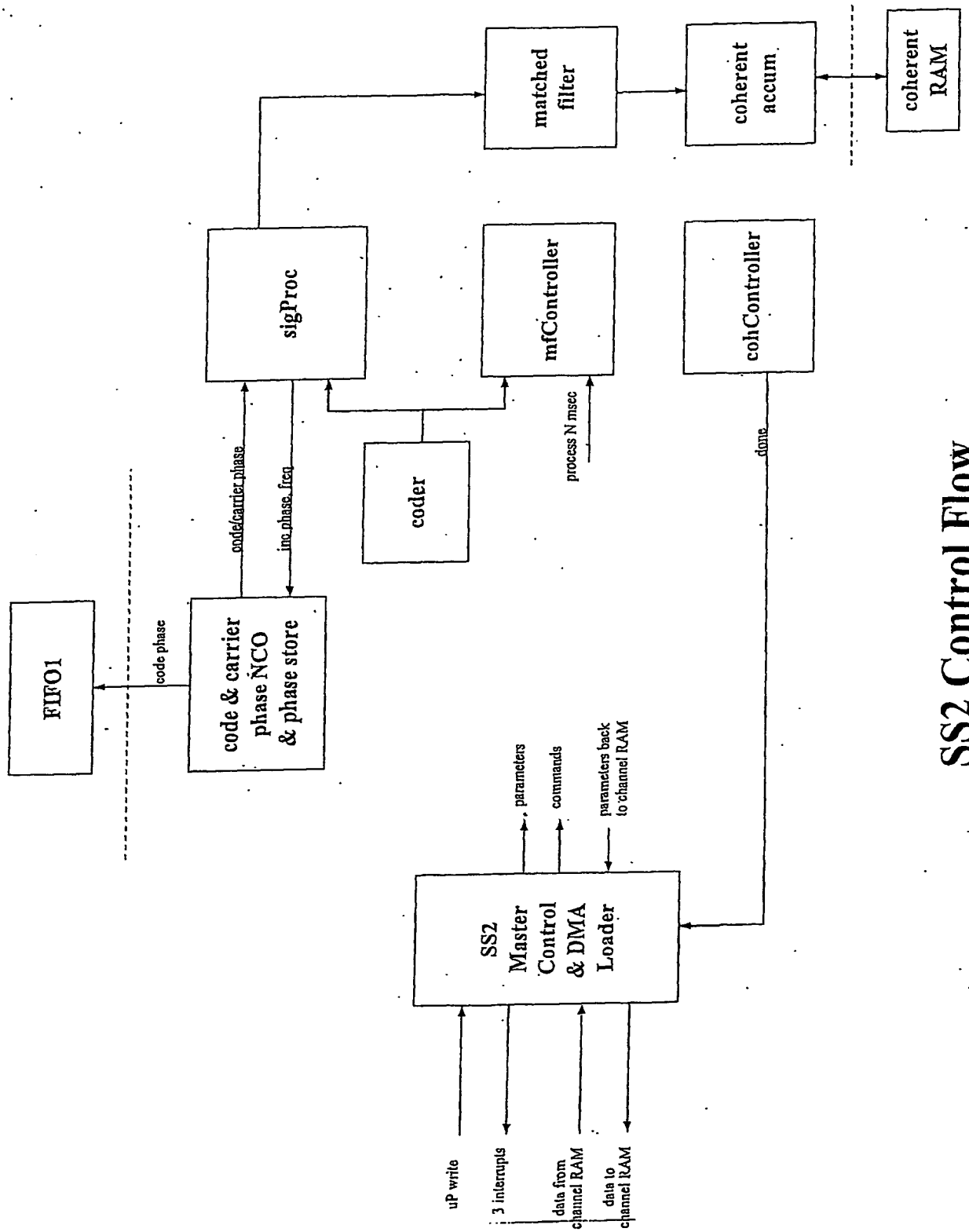


subsystem 2 data flow control



F I G. 1 1

subsystem 2 data flow control



SS2 Control Flow

FIG. 12

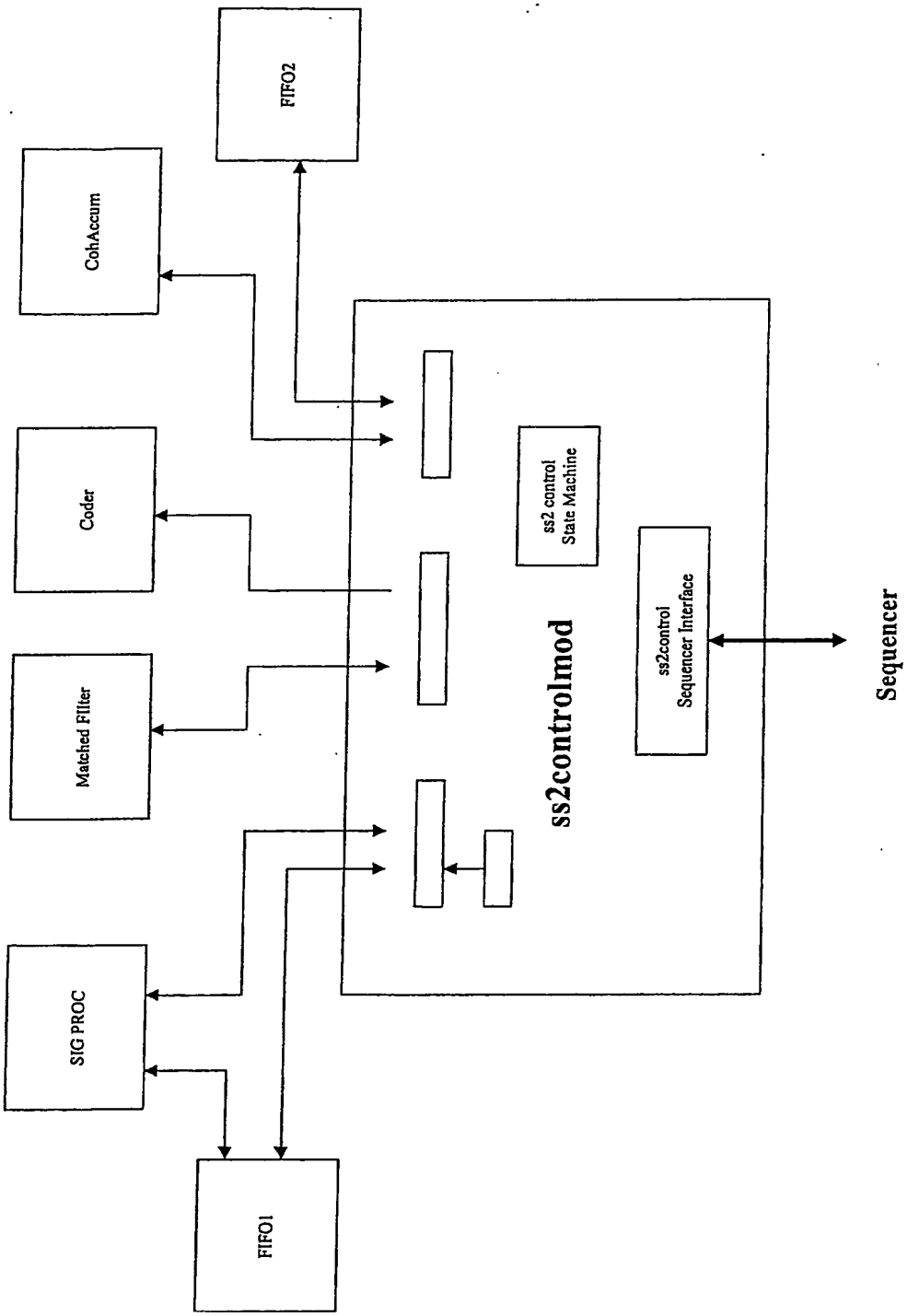
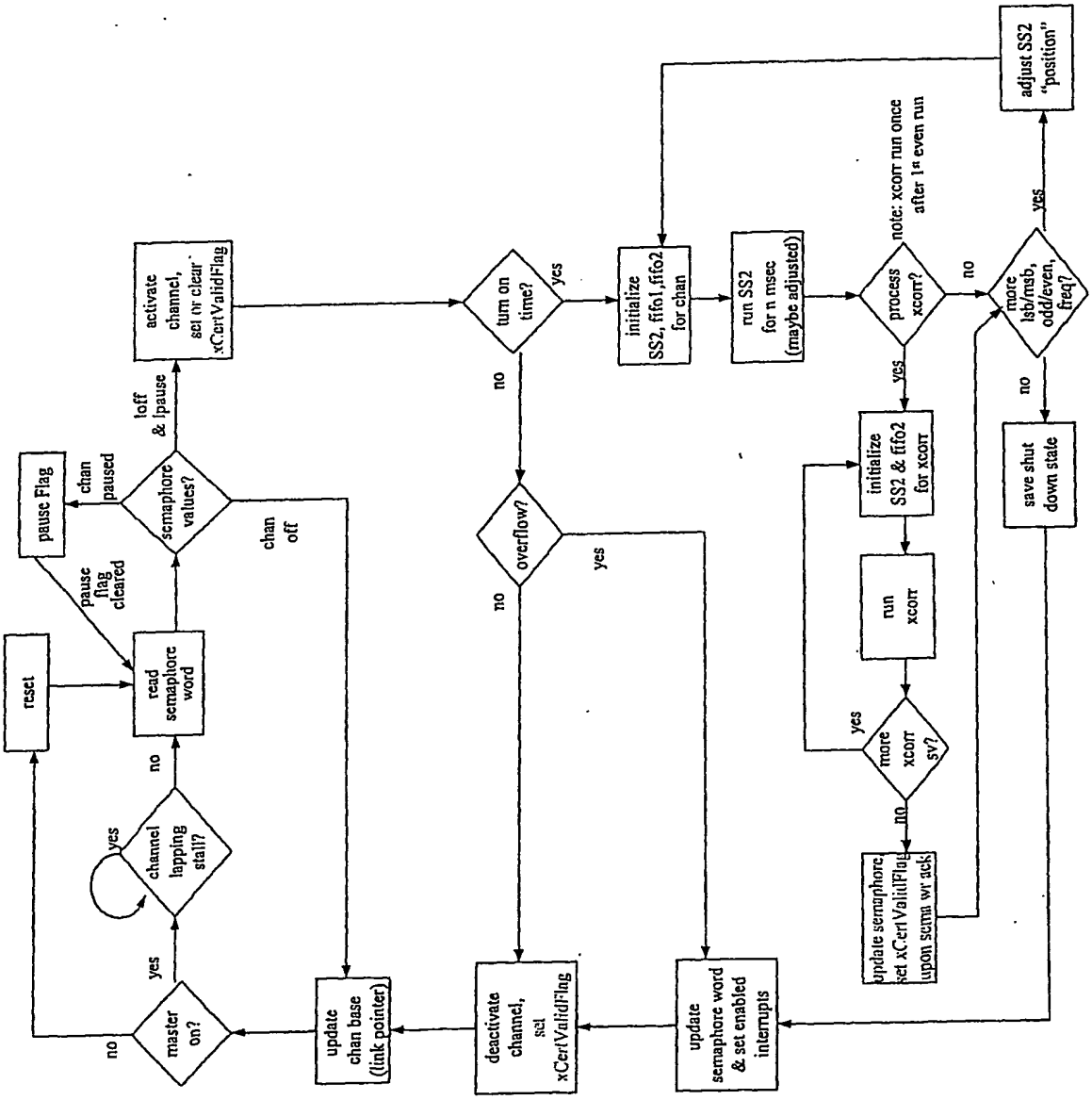


FIG. 13

ss2controlmod interface



SS2 master controller flow — Fig. 14

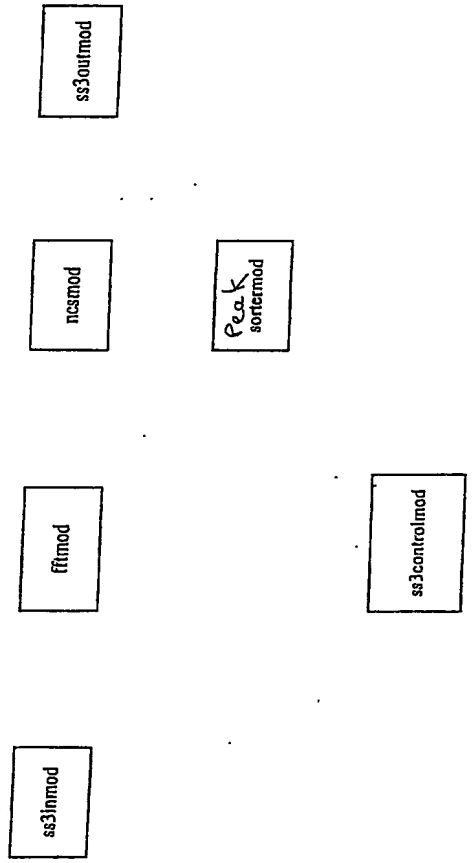


FIG. 15

fft subsystem partitioning

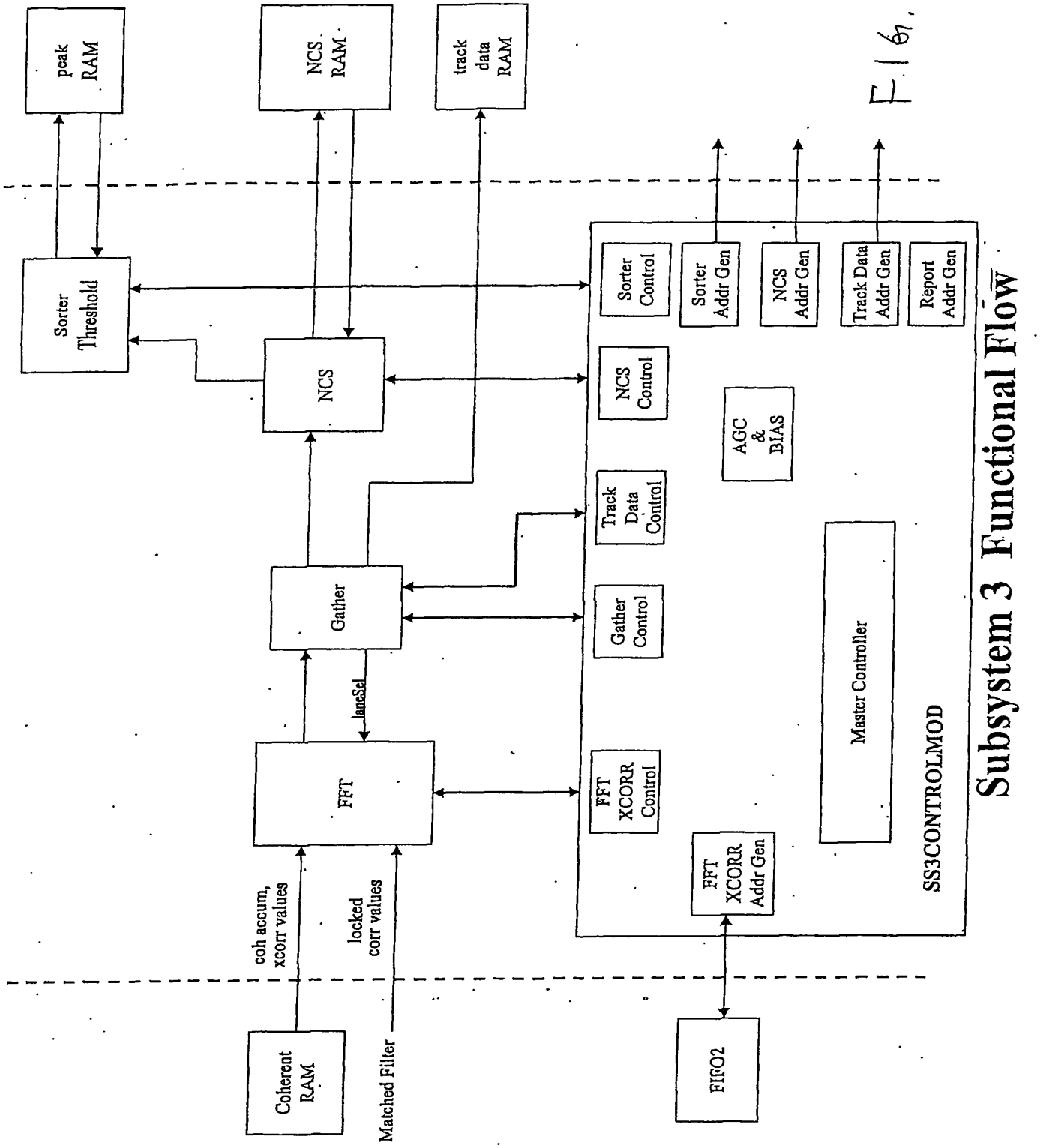

peak RAM
NCS RAM
track data RAM
Sorter Threshold
NCS
Gather
FFT
laneSel
coh accum, xcorr values
locked corr values
Coherent RAM
Matched Filter
FIFO2
Sorter Control
Sorter Addr Gen
NCS Addr Gen
Track Data Addr Gen
Report Addr Gen
NCS Control
AGC & BIAS
Track Data Control
Gather Control
FFT XCORR Control
FFT XCORR Addr Gen
Master Controller
SS3CONTROLMOD
Subsystem 3 Functional Flow

Fig. 16

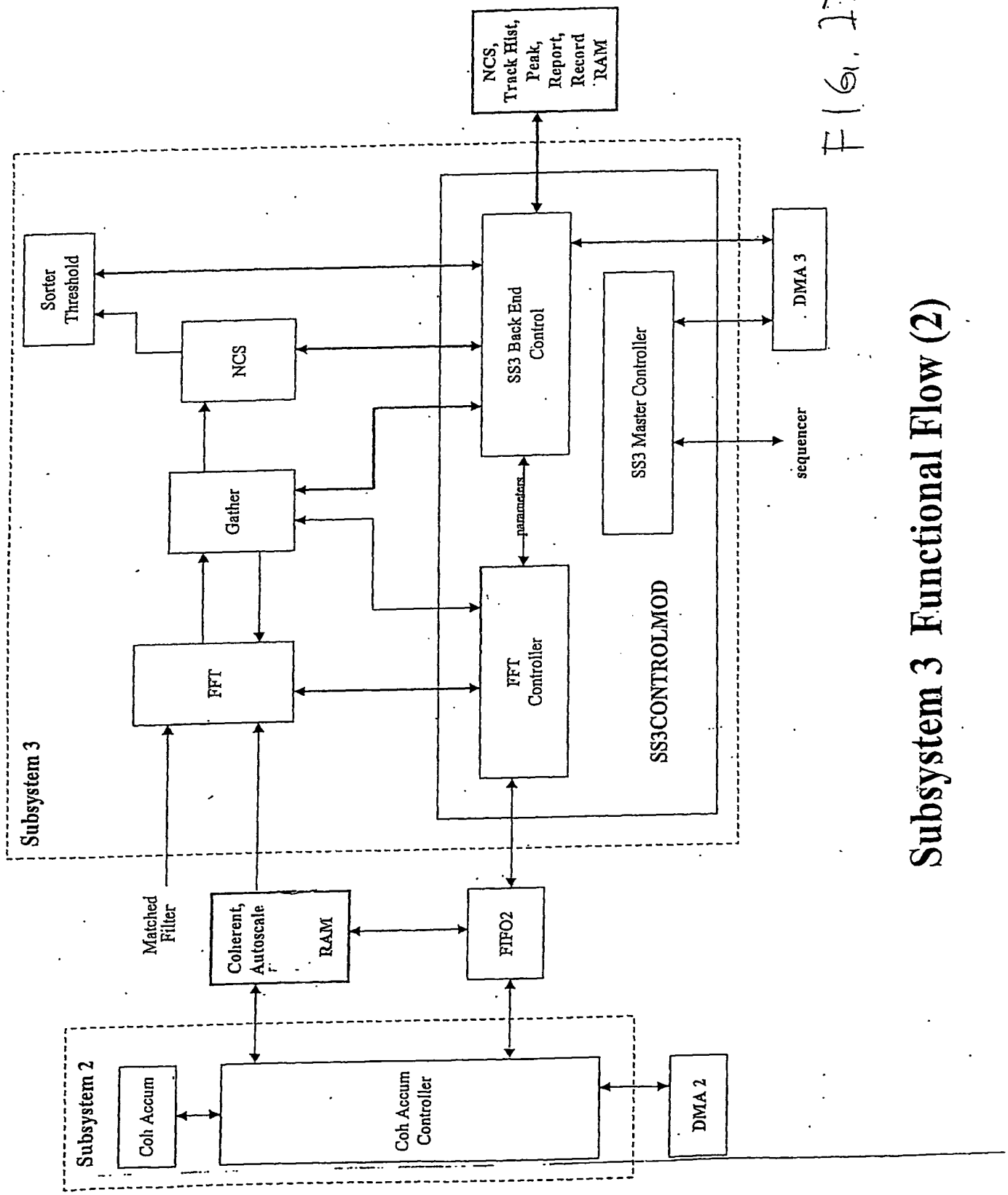
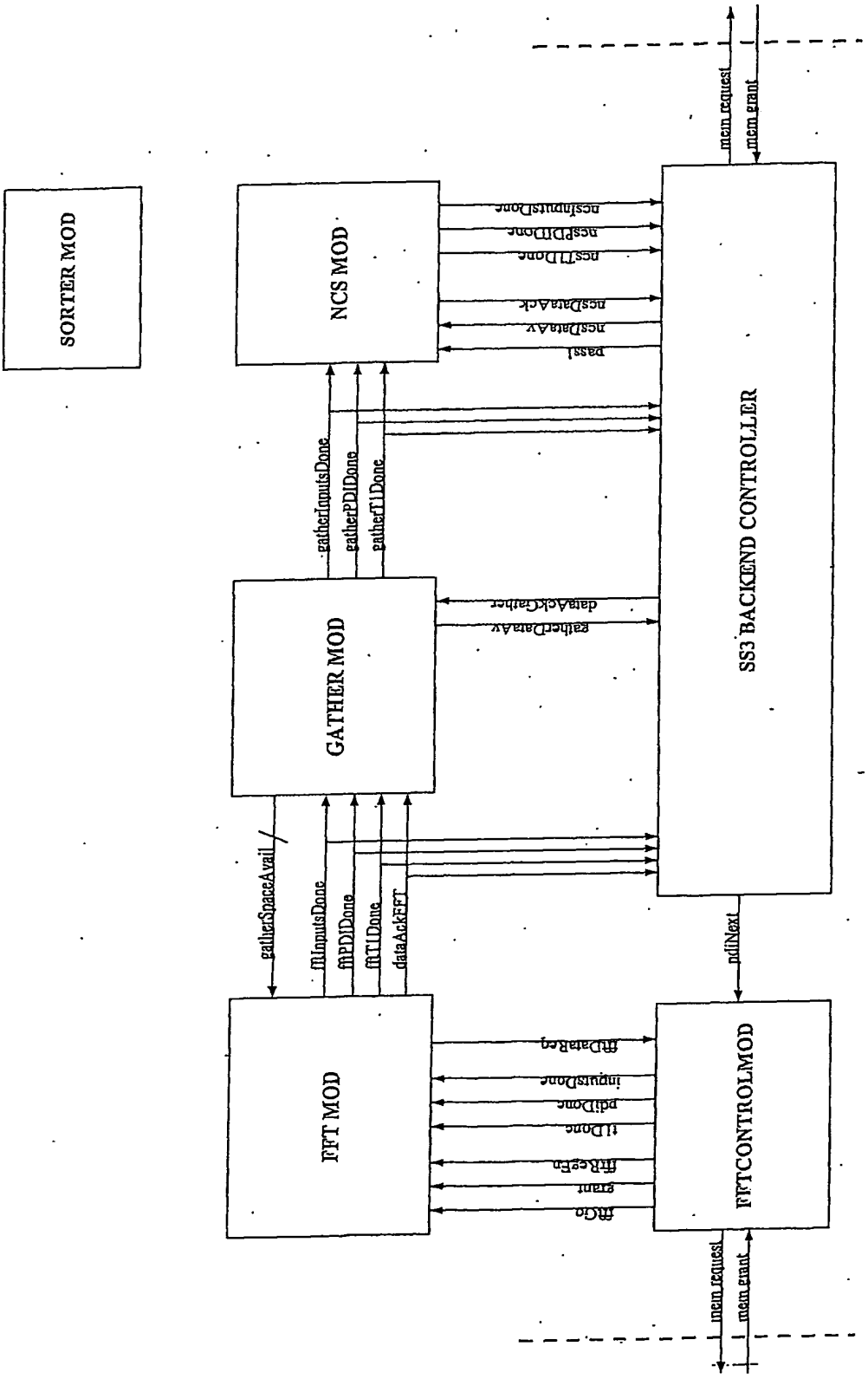


Fig. 17

Subsystem 3 Functional Flow (2)



F I G. 18

Subsystem 3 Control Flow

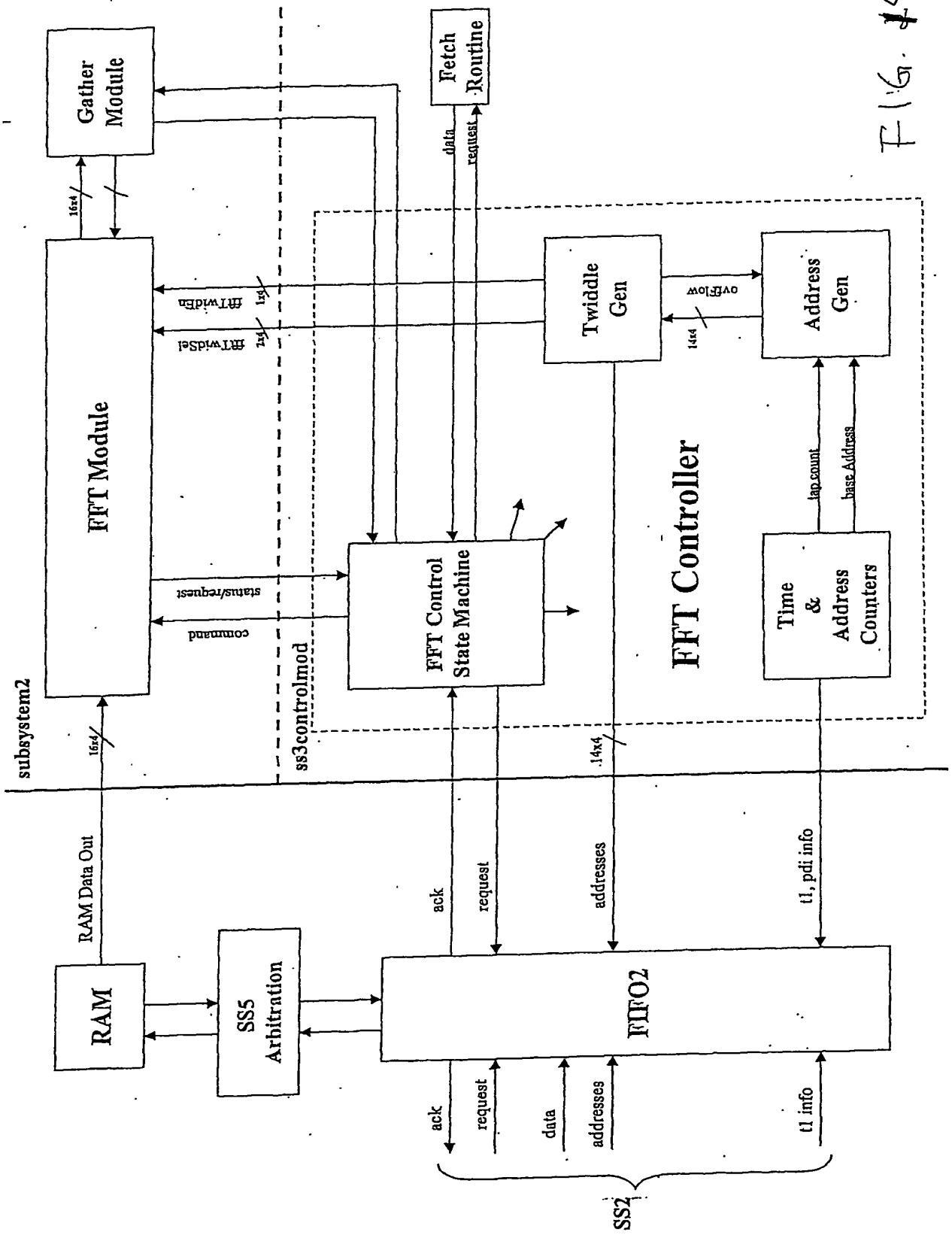


FIG. 19

FFT Controller Flow

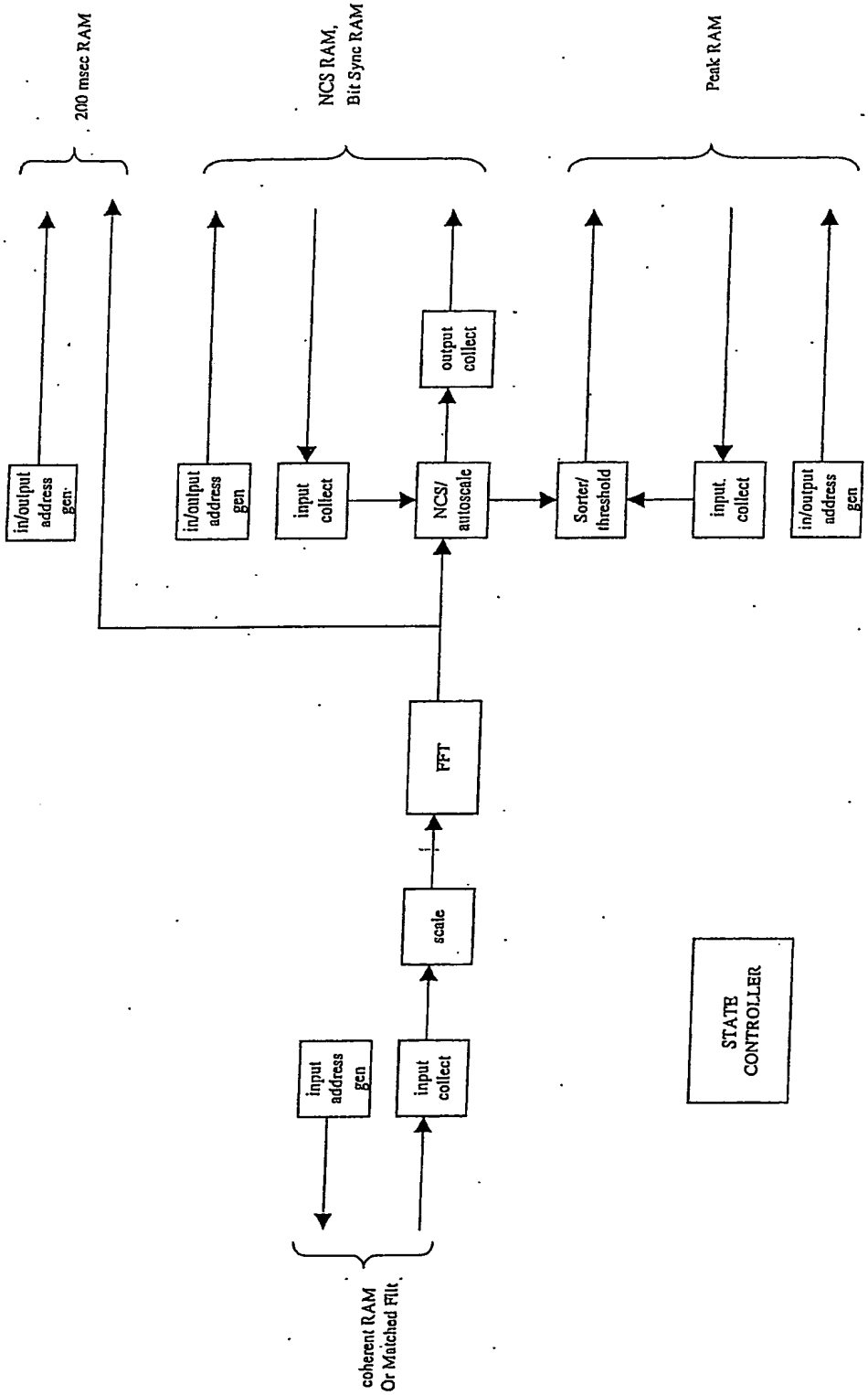
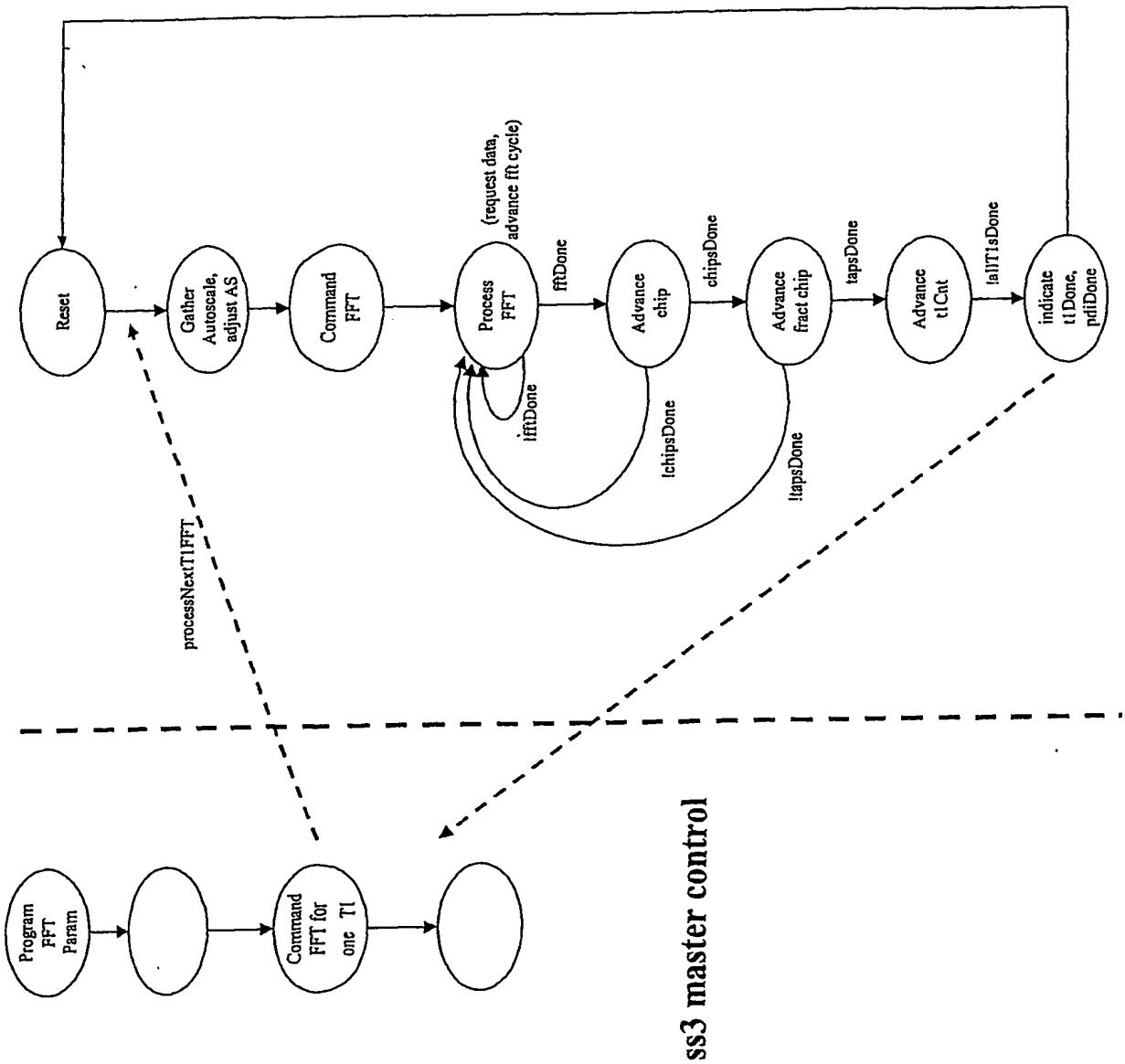
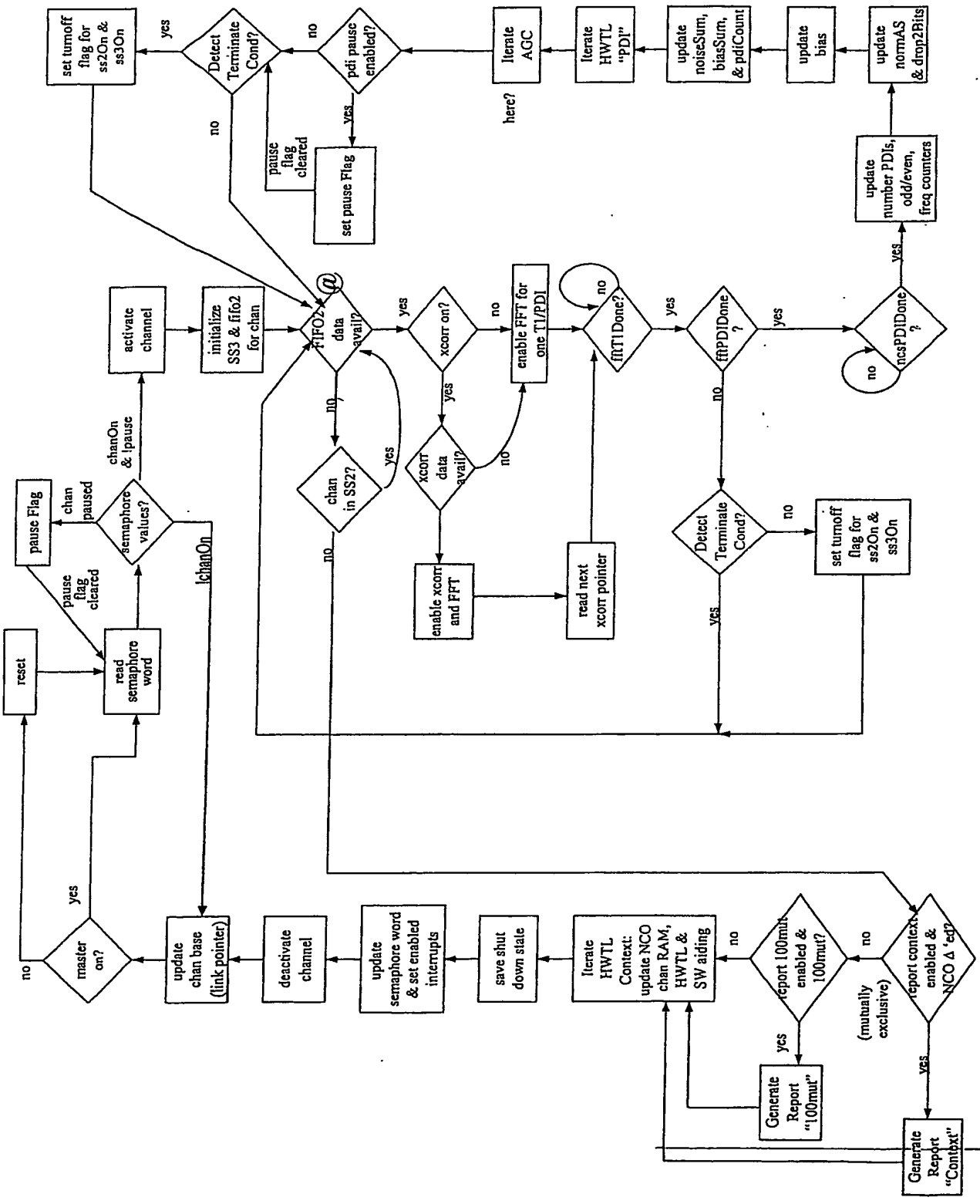


FIG. 20

fft subsystem flow

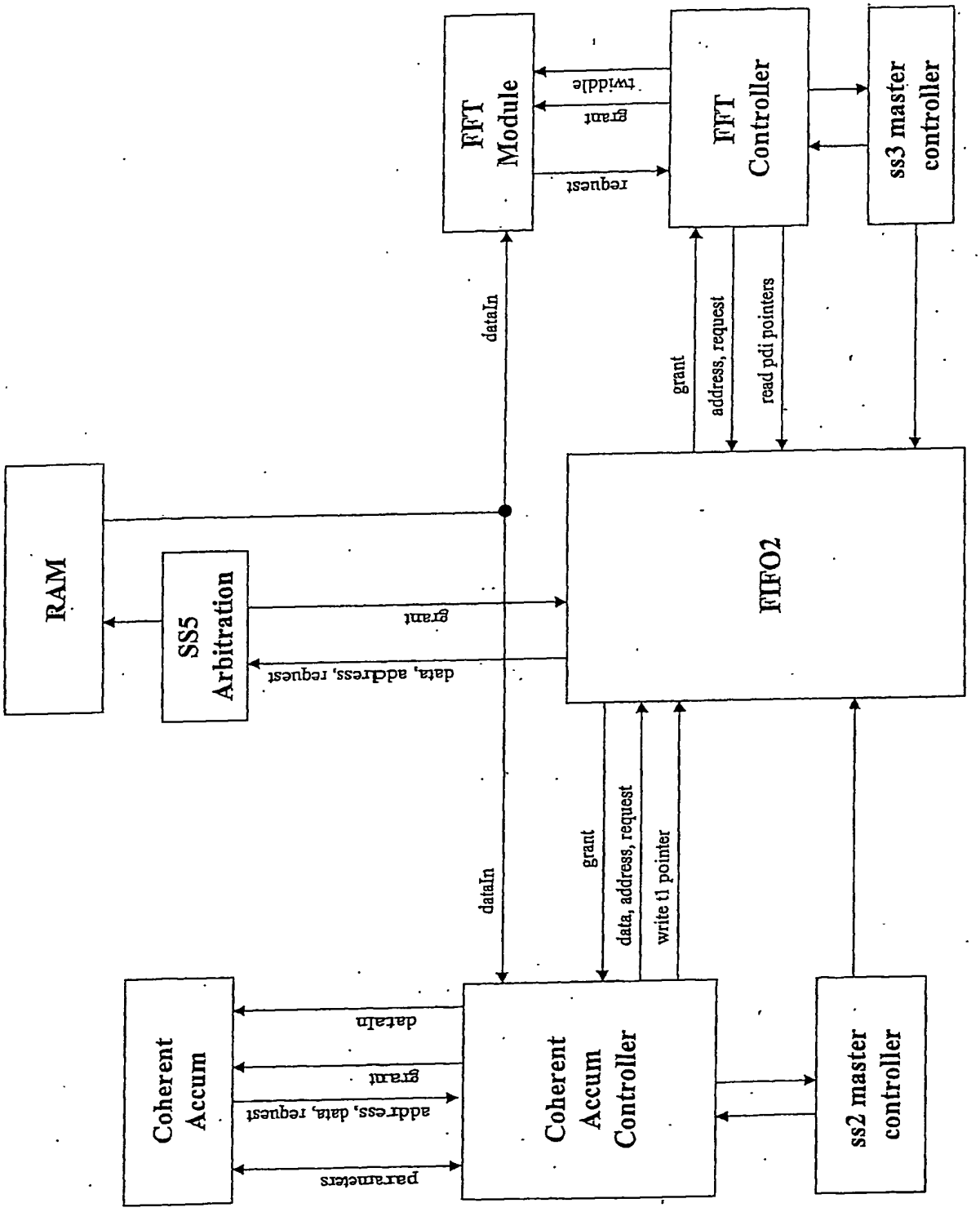

Reset
Gather Autoscale, adjust AS
Command FFT
Process FFT
(request data, advance fft cycle)
fftDone
Advance chip
chipsDone
Advance fract chip
tapsDone
Advance t1Cnt
lastT1sDone
indicate t1Done, pdiDone
!fftDone
!chipsDone
!tapsDone
processNextT1FFT
Program FFT Param
Command FFT for one T1
ss3 master control
FFT Controller Flow
FFT controller

F16. 21



SS3 master controller flow

Fig 22



Subsystem2 / FIFO2 / Subsystem3 Flow

FIG. 23

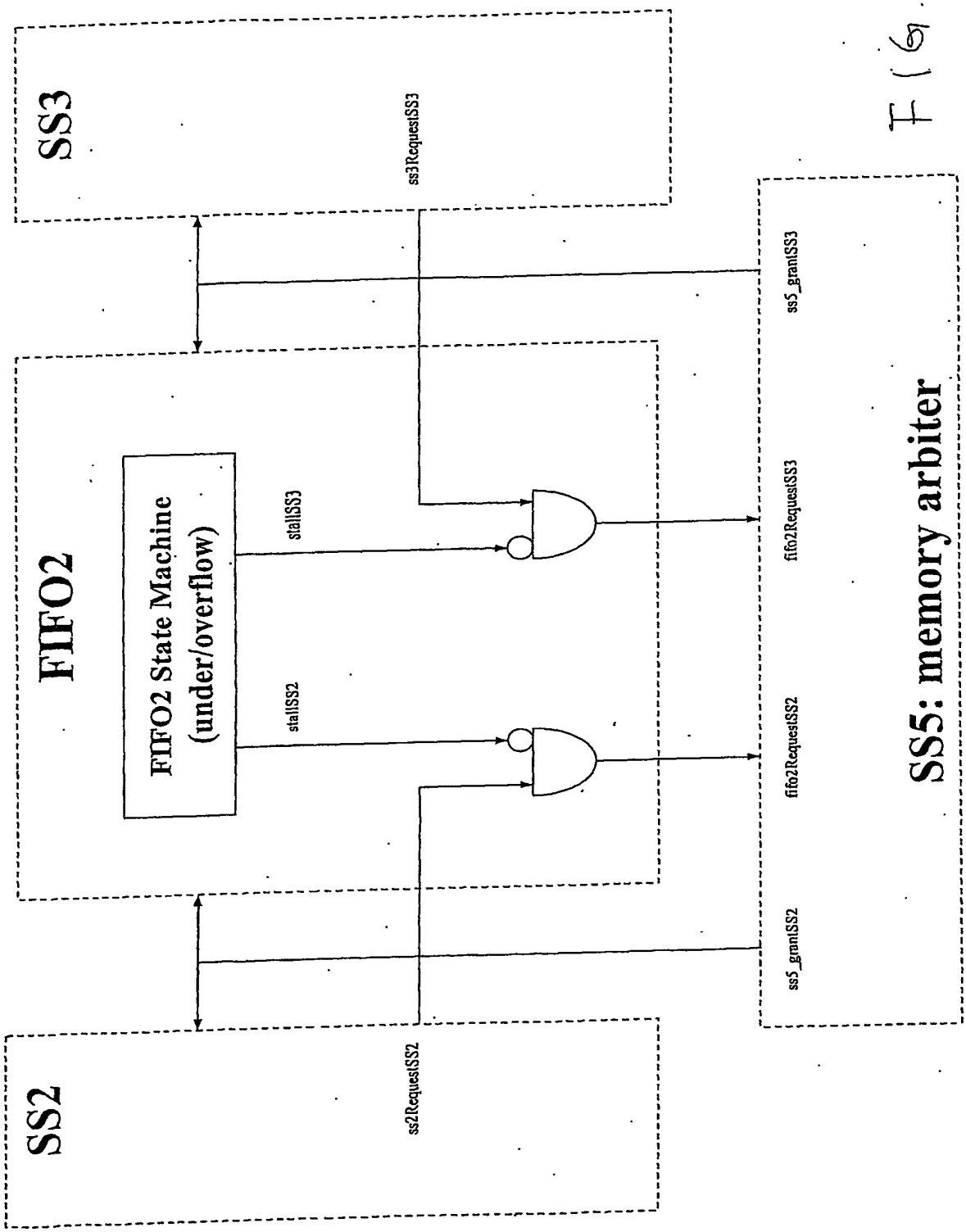

SS3
ss3RequestSS3
FIFO2
FIFO2 State Machine
(under/overflow)
stallSS3
stallSS2
SS2
ss2RequestSS2
ss5_grantSS3
fifo2RequestSS3
fifo2RequestSS2
ss5_grantSS2
SS5: memory arbiter

Fig. 24

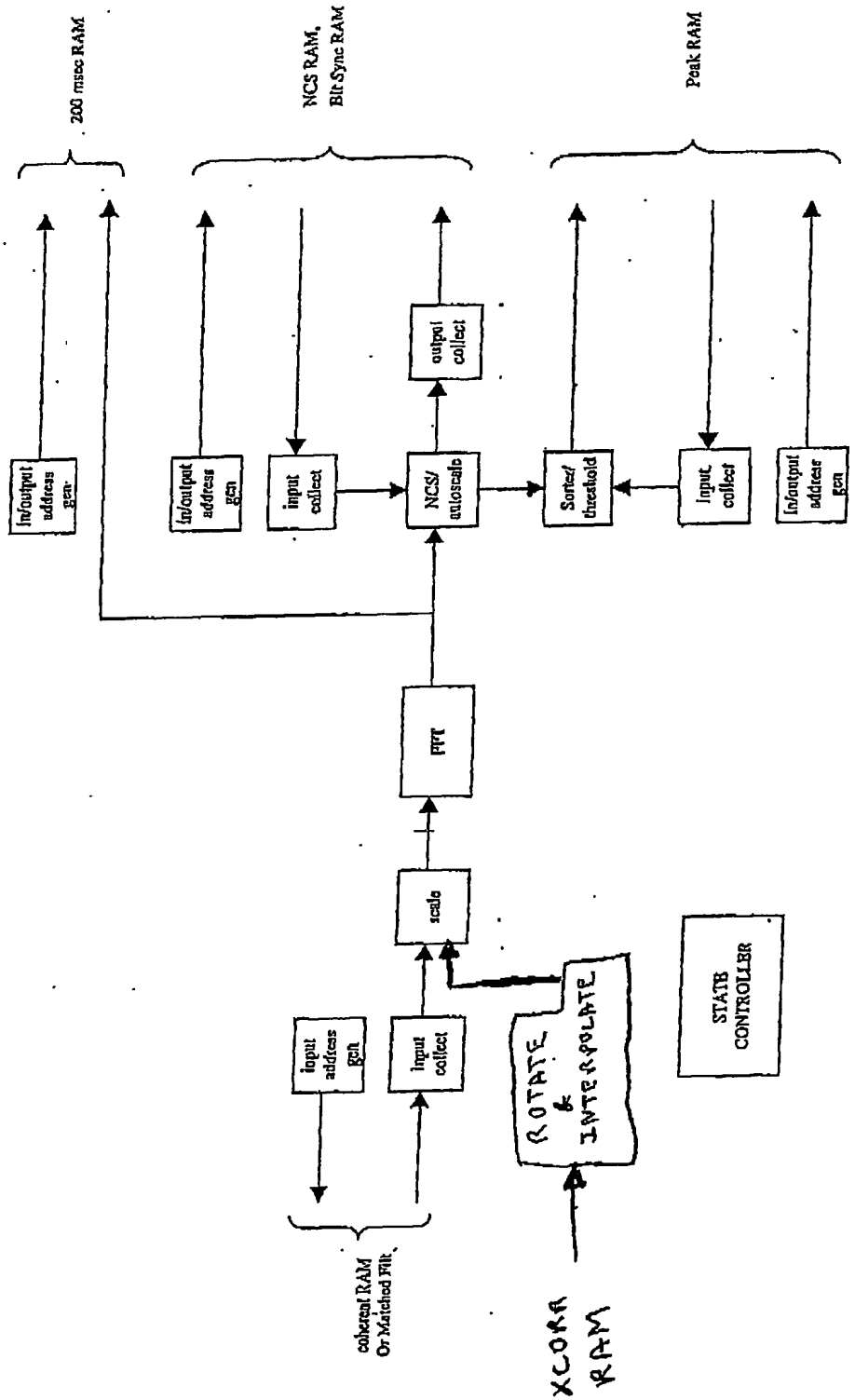
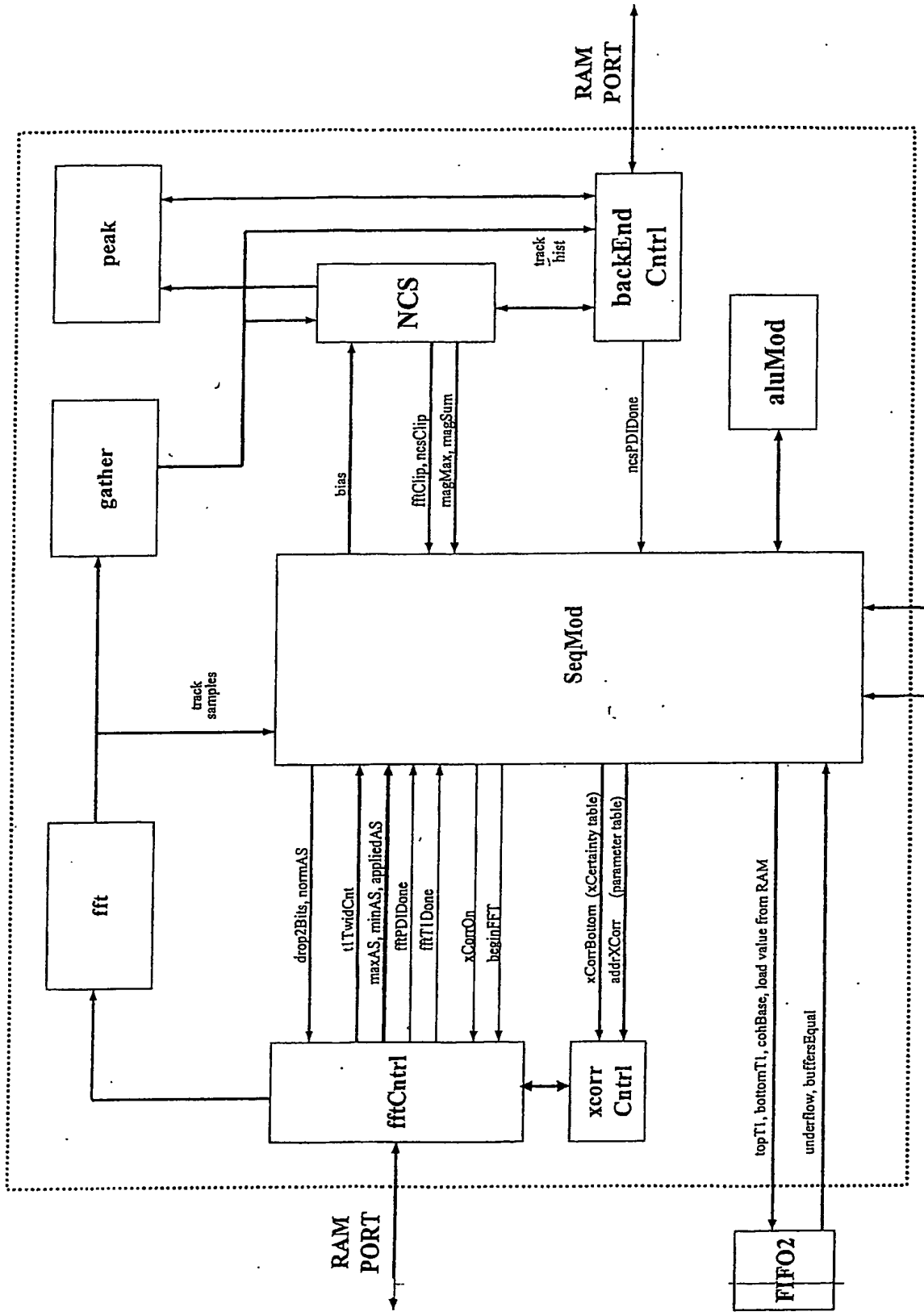


FIG. 25

fft subsystem flow

FIG. 26

sequencer subsystem partitioning



Sequencer Module Interface

F 16, 27

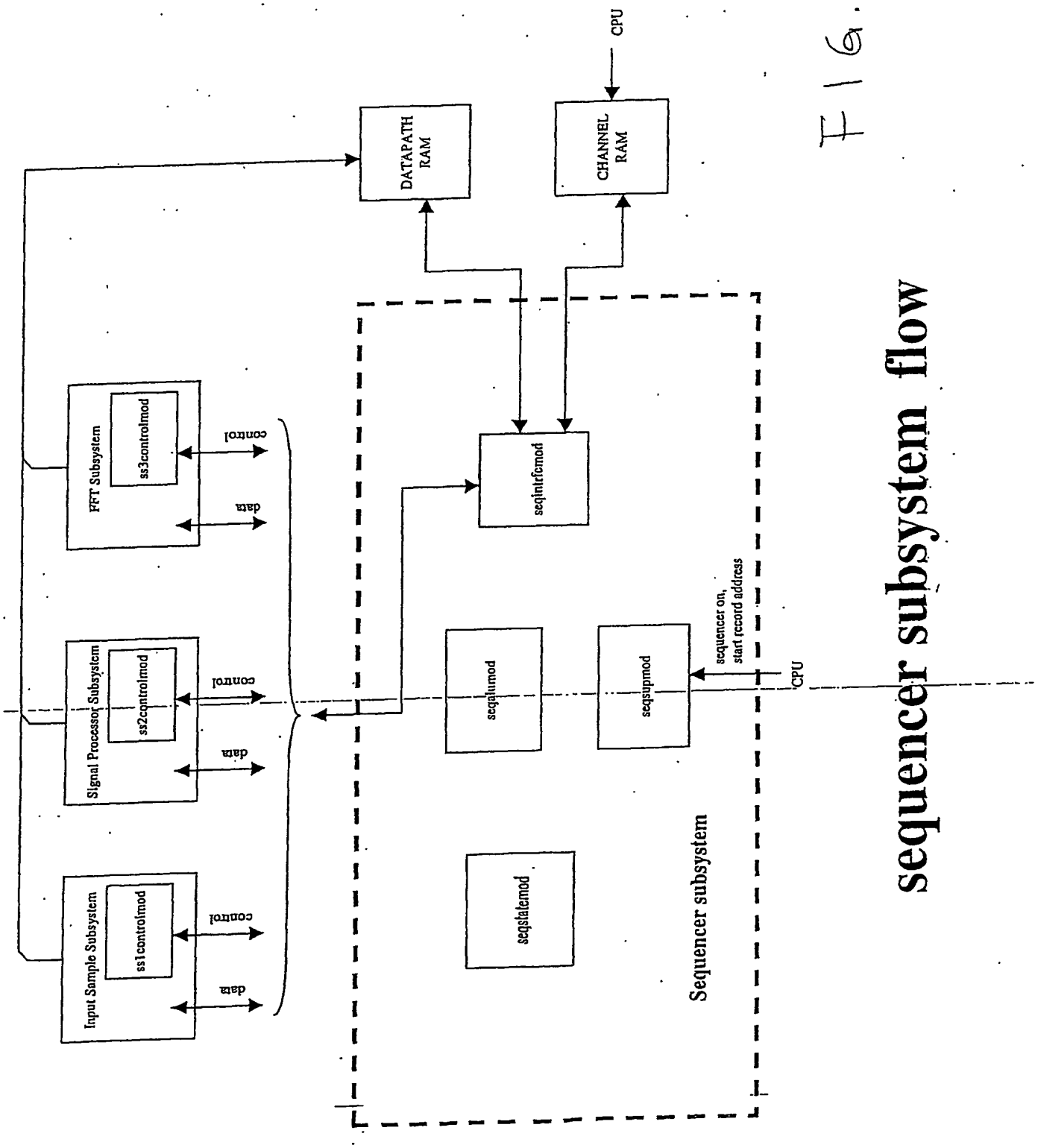


FIG. 28

sequencer subsystem flow

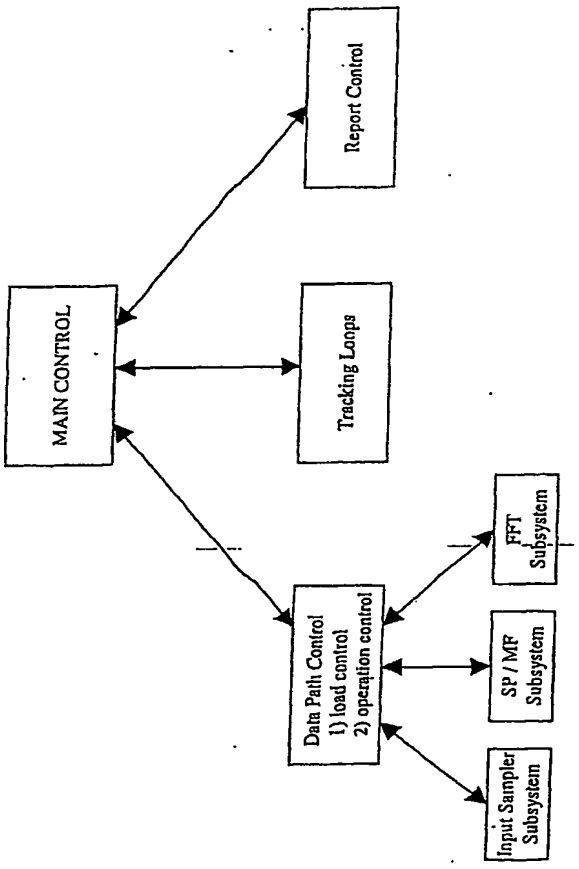


FIG. 29

sequencer state machine flow

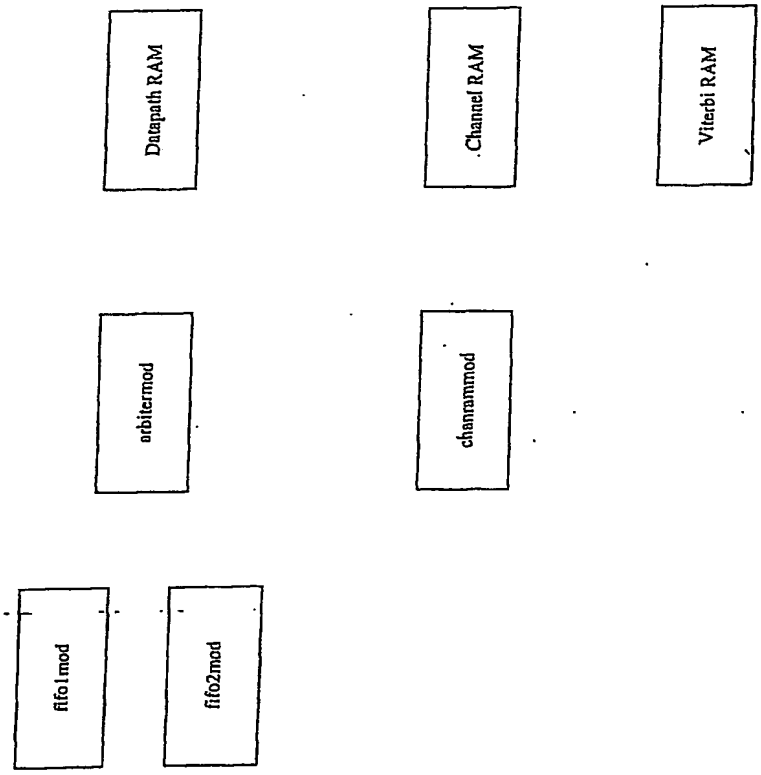

memory subsystem
Datapath RAM
Channel RAM
Viterbi RAM
arbitermod
chanrammod
fifo1mod
fifo2mod

FIG. 30

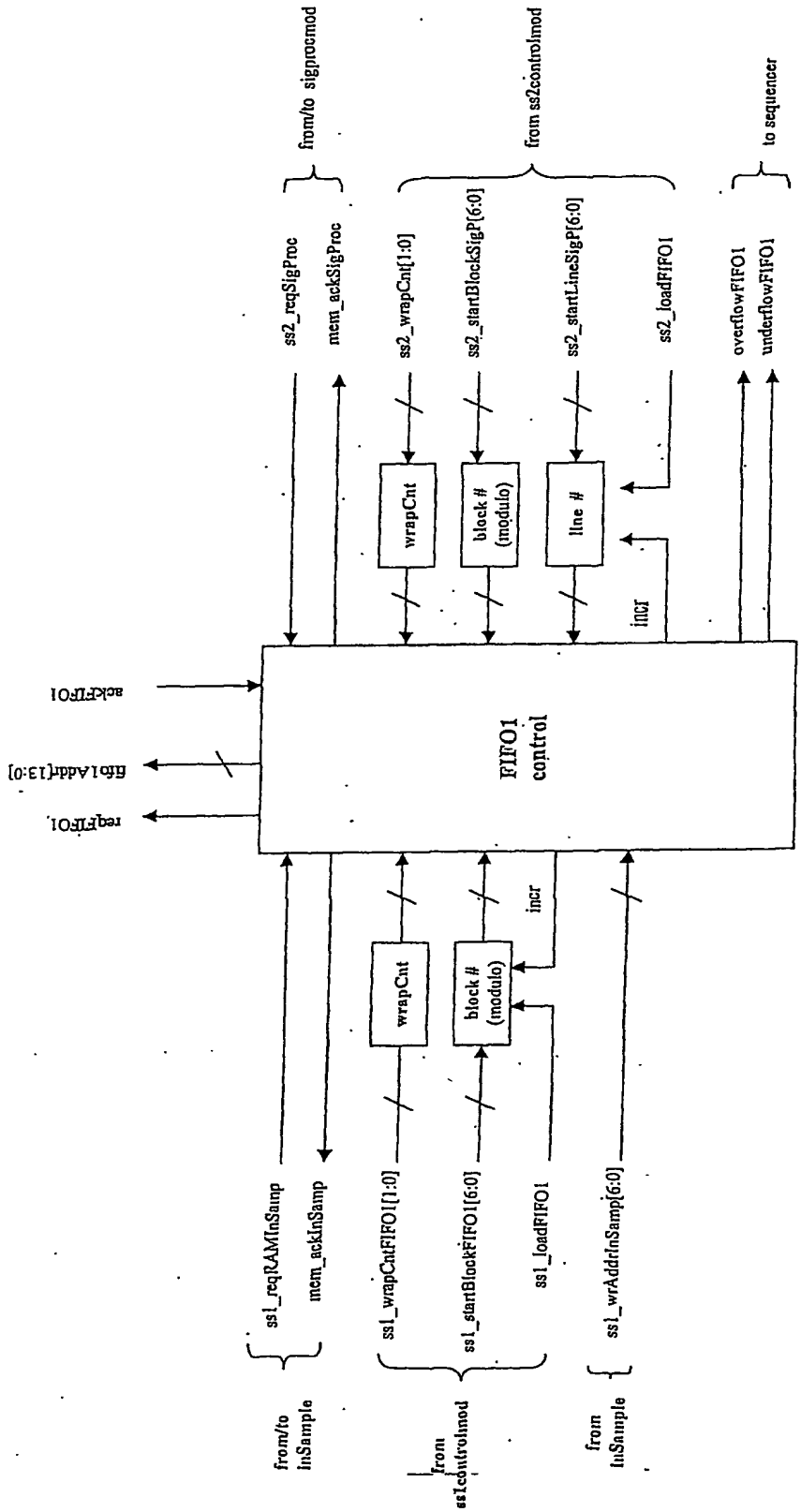


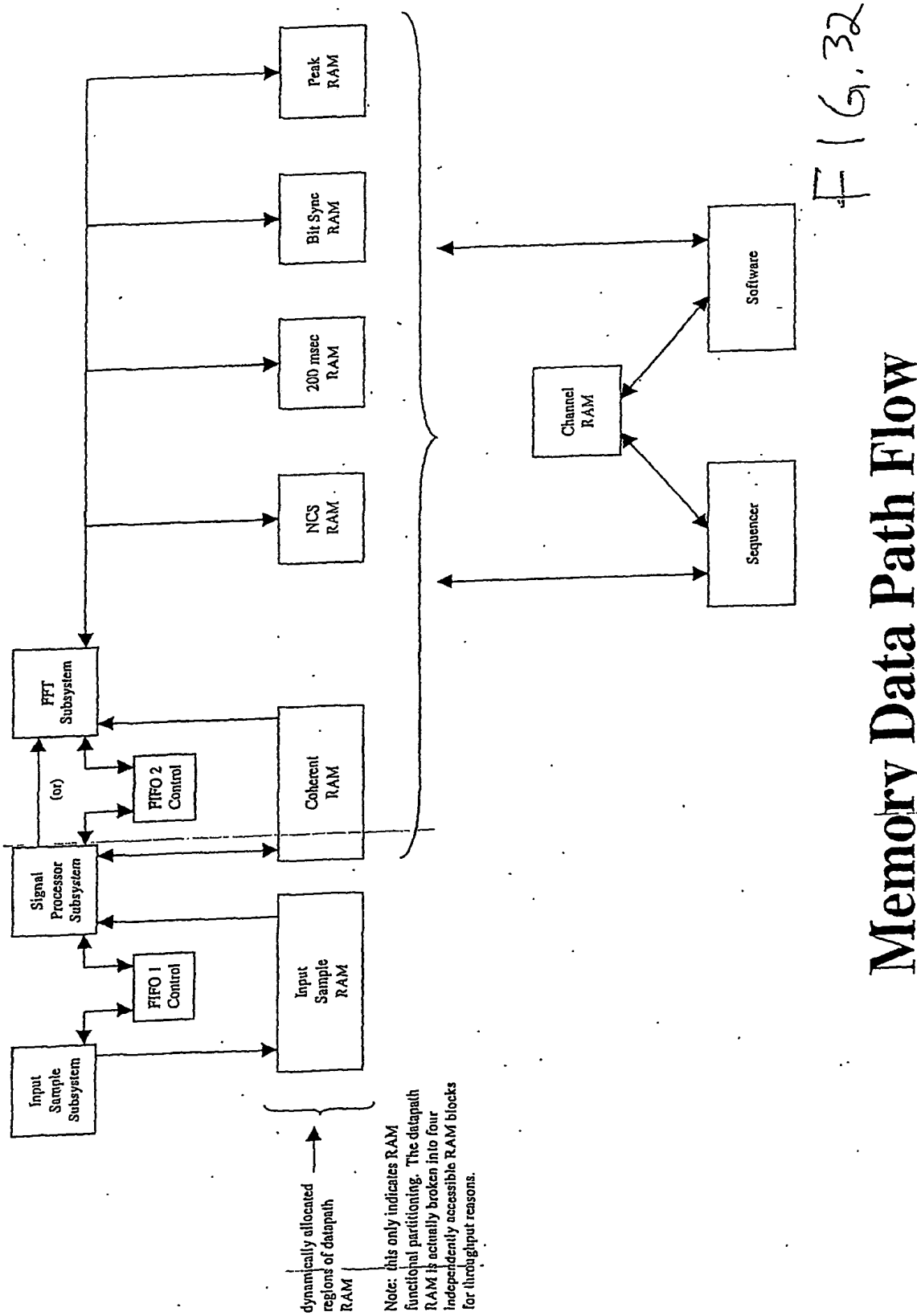
FIG. 31

fifo1 structure

Note:

ramp rate	line	mSec
16 samp/chip	2 chips/line	512 lines/mSec
4 samp/chip	8 chips/line	128 lines/mSec

128 lines / block



Memory Data Path Flow

F16.32

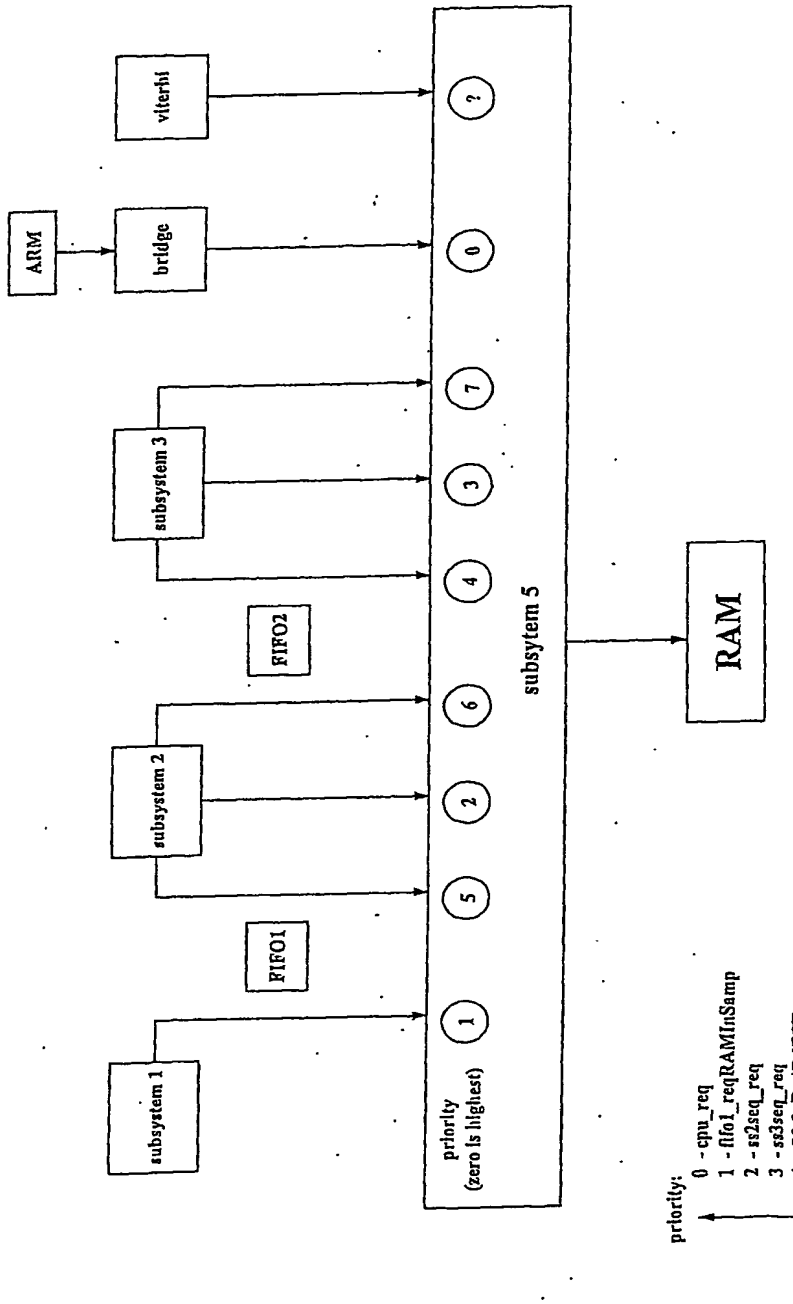
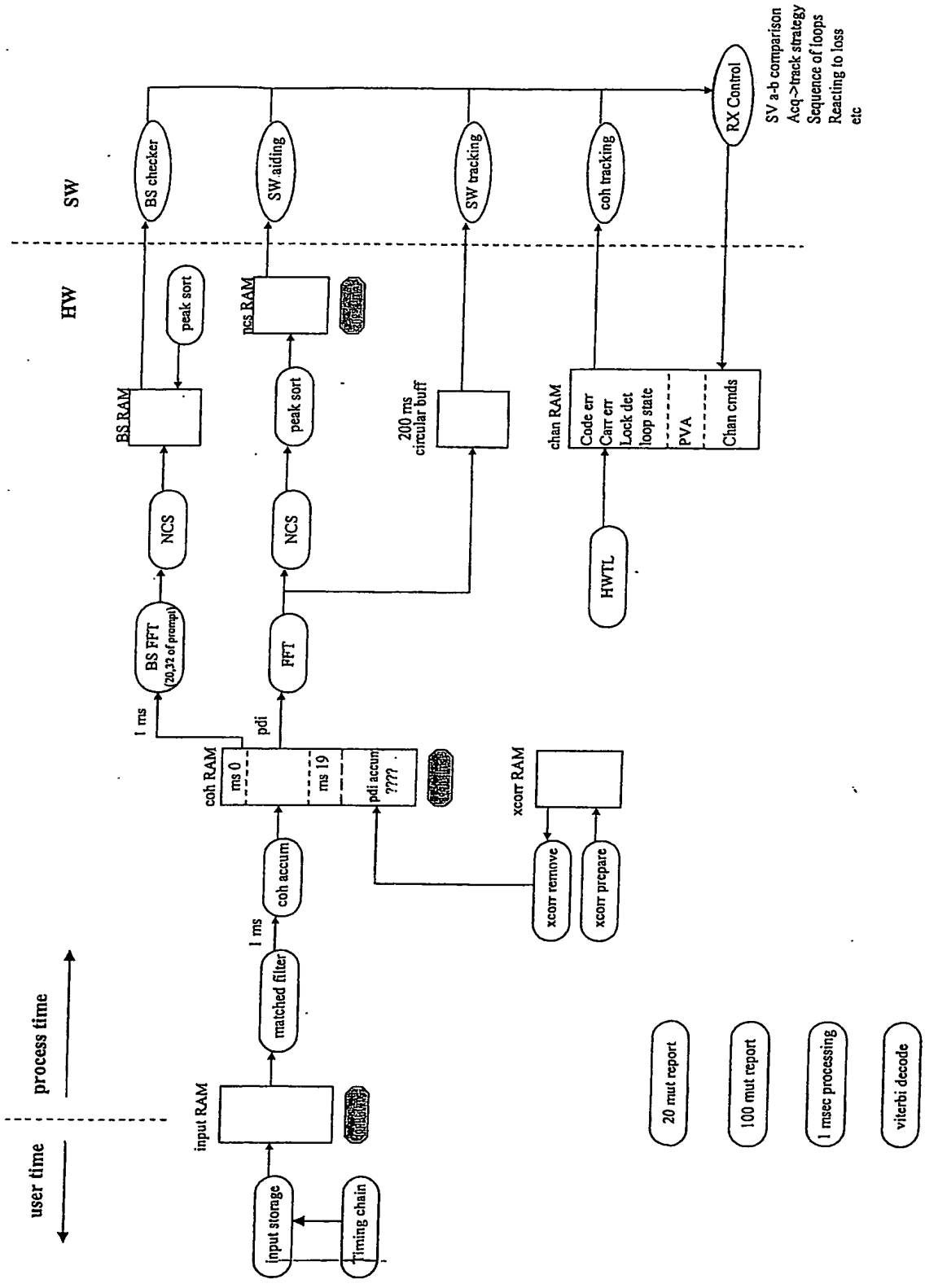


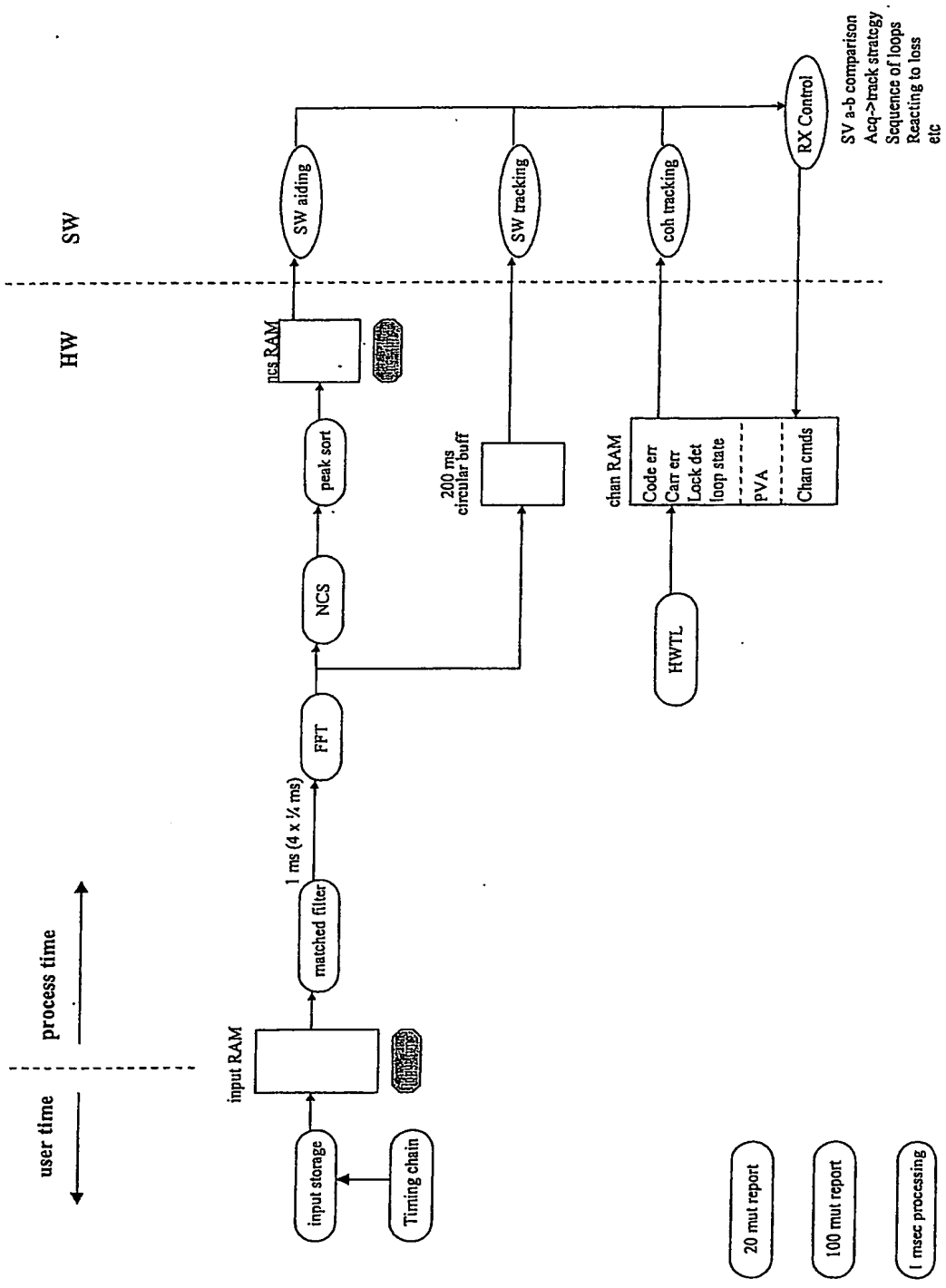
FIG. 33

ss5 arbitration priority



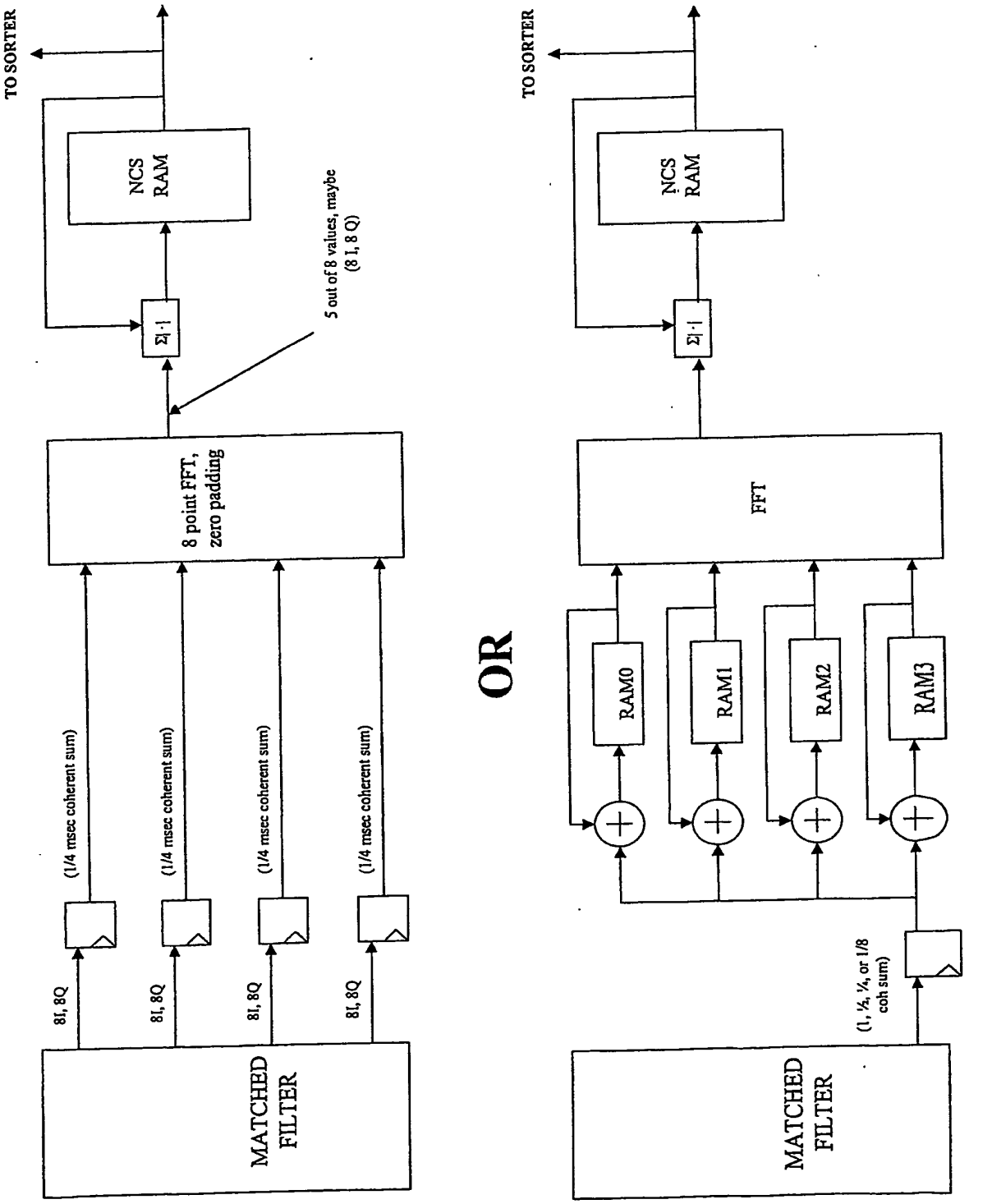
1 msec acq / 1/8 msec track mode process flow

F16, 34



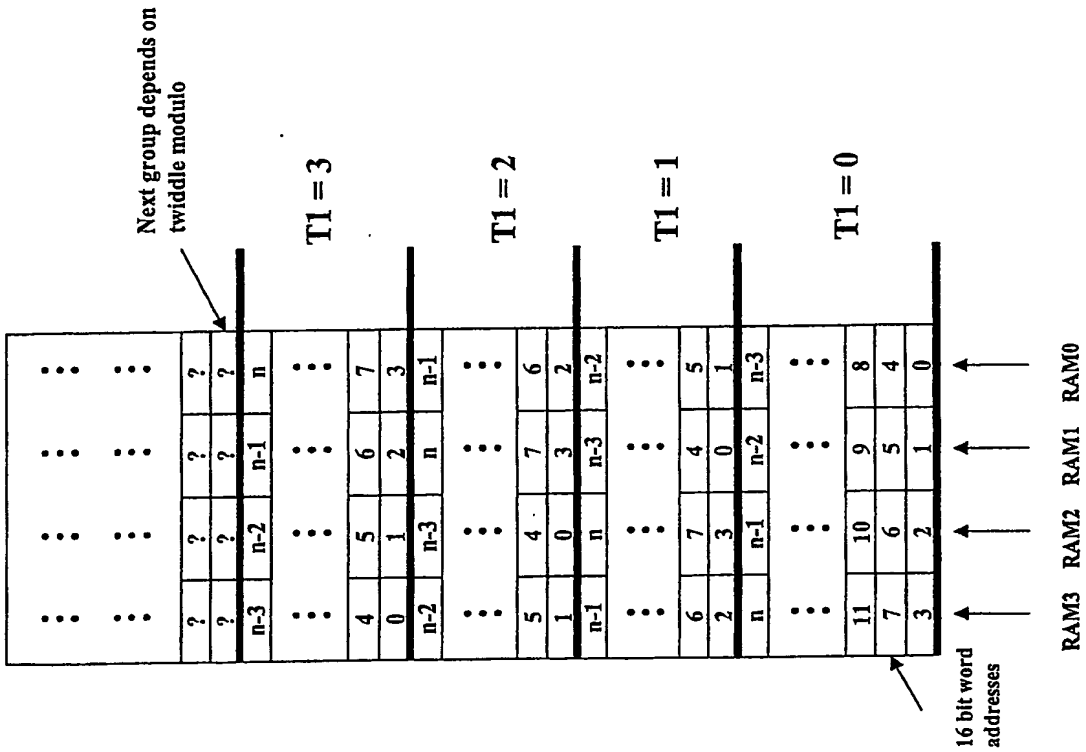
cold start acquisition mode process flow

F16.35



CORRELATOR DATA PATH

FIG. 36



Coherent RAM Data Storage

FIG. 37.

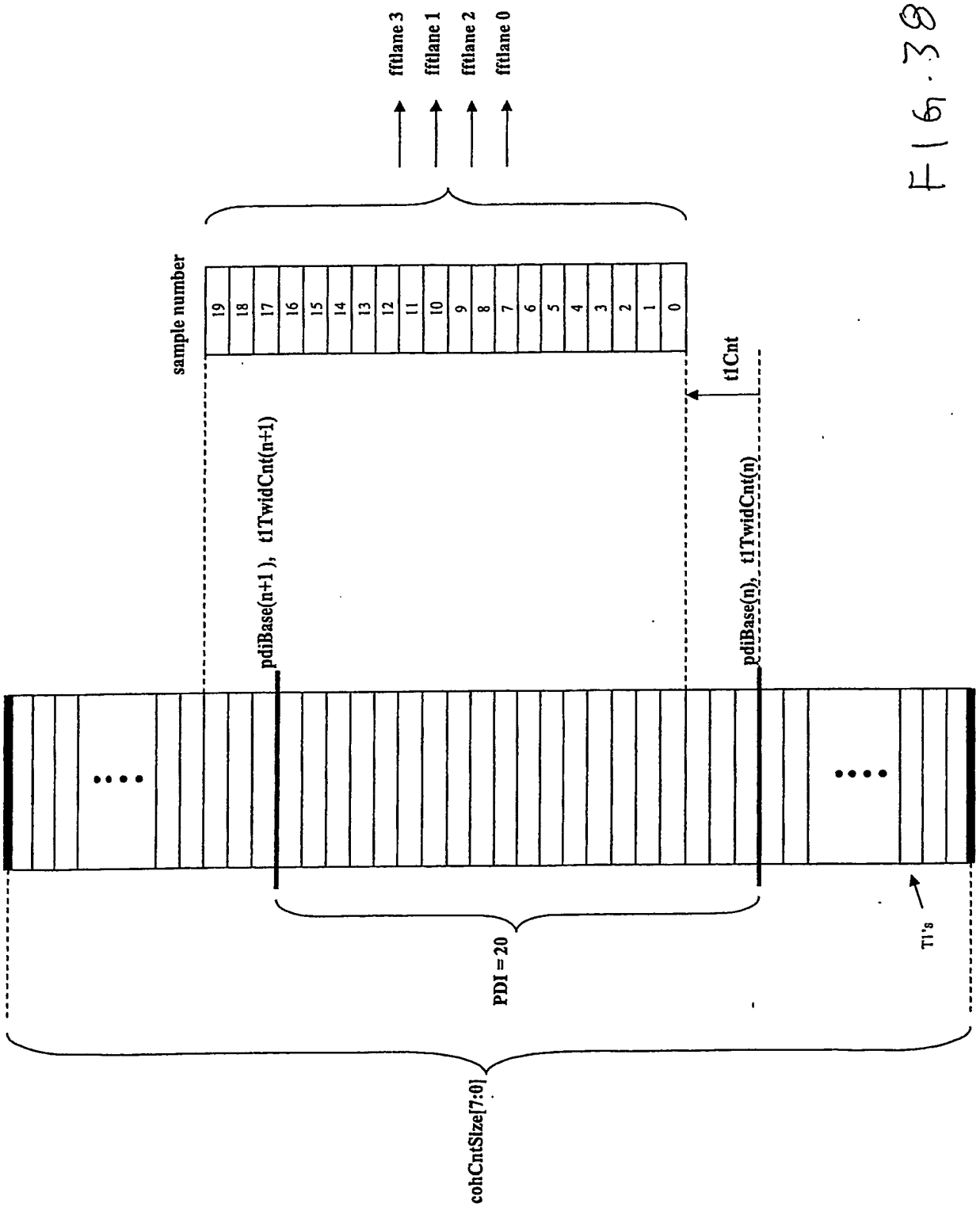


FIG. 38

FFT Access of Coherent RAM (eg. pdi = 20)

fftMode = 0 {4,8} Stride=1, Twiddle Modulo = 4
0,2,1,3

fftMode = 1 {8,8} Stride=2, Twiddle Modulo = 8
or fftMode = 2 {8,16}
0,4,2,6
1,5,3,7

fftMode = 3 {16,16} Stride=4, TwiddleModulo = 16
or fftMode = 4 {16,32}
0,4,8,c
1,9,5,d
2,a,6,e
3,b,7,f

fftMode = 5 {20,32} Stride=4, Twiddle Modulo = 16
10,12,11,13 - may have collisions
00,08,04,0c
01,09,05,0d
02,0a,06,0e
03,0b,07,0f

FFT - Order Of Data Needed

FIG. 39

twidModSel = 2, modulo 16 :

twiddled positon = (t1Twid[3:2] ^ t1Twid[1:0] ^ tapPostion[1:0])

twiddleSel = (t1Twid[3:2] ^ t1Twid[1:0])

twidModSel = 1, modulo 8 :

twiddled positon = ((1'b0 , t1Twid[2]) ^ t1Twid[1:0] ^ tapPostion[1:0])

twiddleSel = ((1'b0 , t1Twid[2]) ^ t1Twid[1:0])

twidModSel = 0, modulo 4 :

twiddled positon = ((1'b0 , 1'b0) ^ t1Twid[1:0] ^ tapPostion[1:0])

twiddleSel = ((1'b0 , 1'b0) ^ t1Twid[1:0])

Coherent RAM Input Twiddle Select

FIG. 40

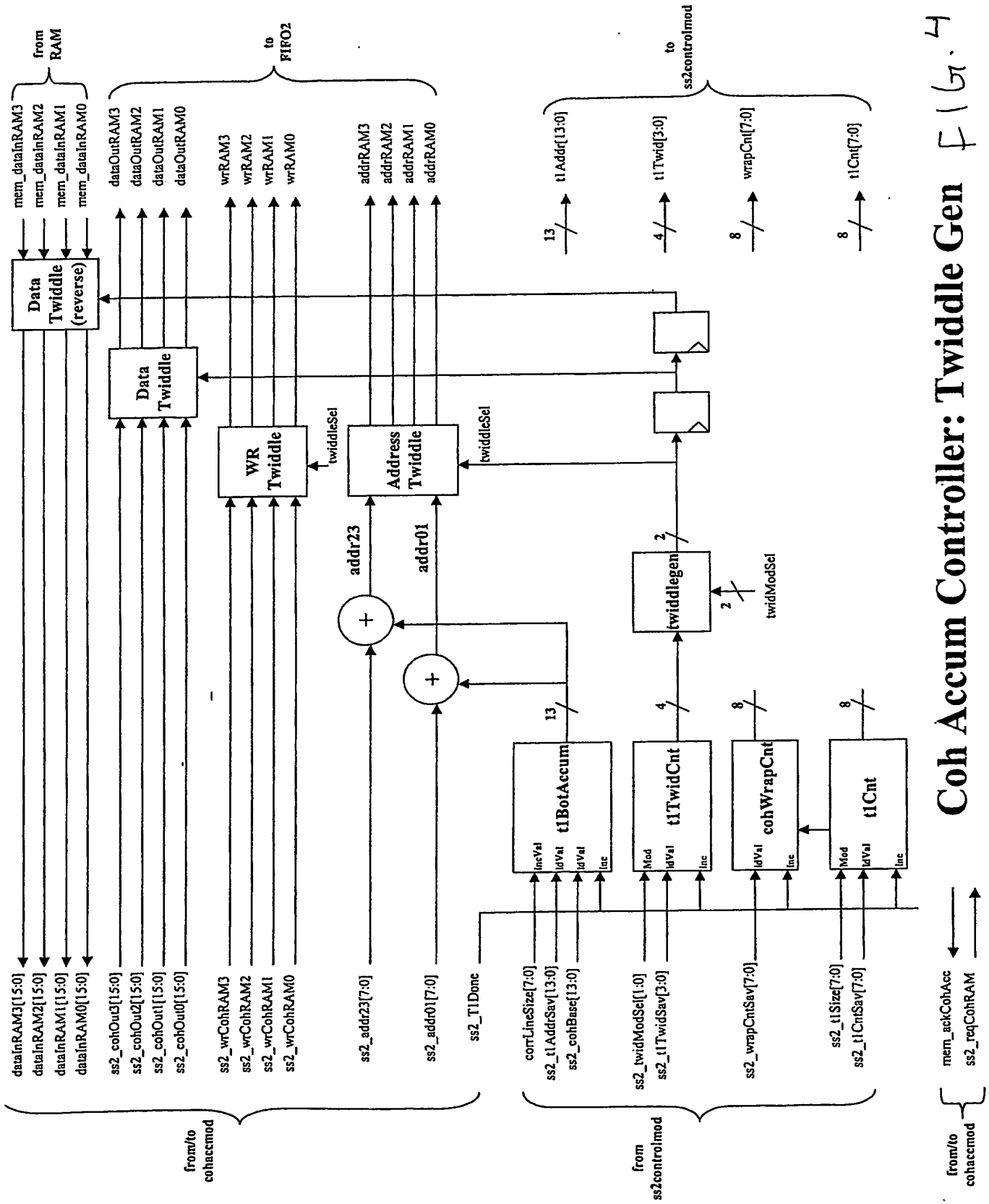
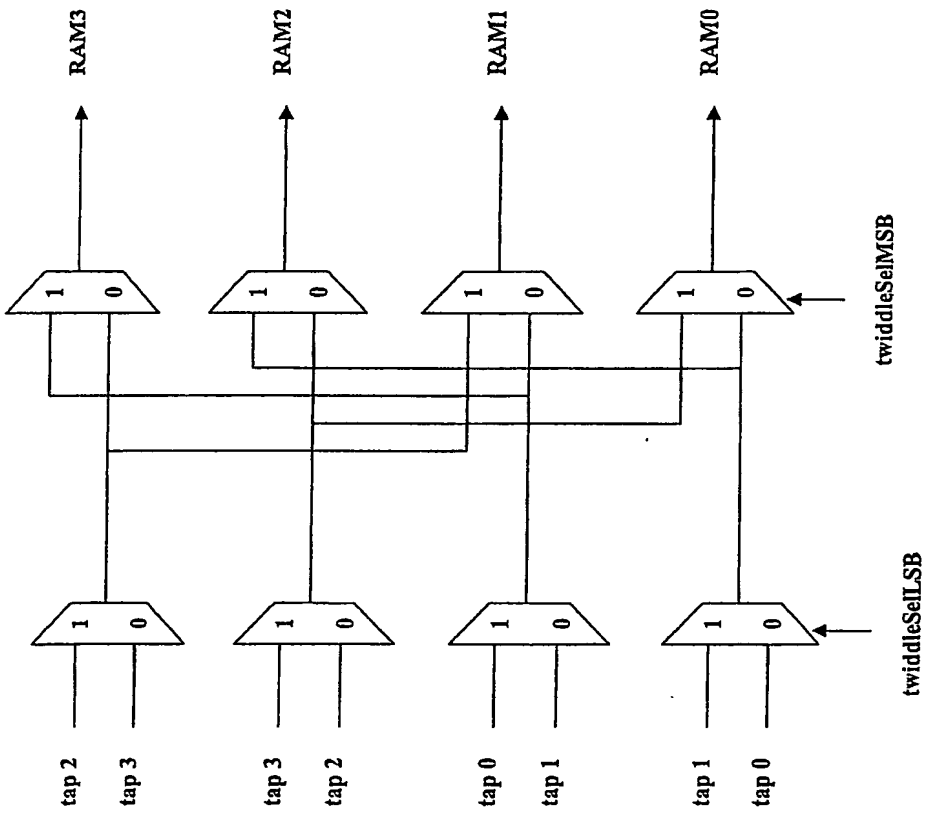

Coh Accum Controller: Twiddle Gen
from RAM
to FIFO2
to ss2controlmod
Data Twiddle (reverse)
Data Twiddle
WR Twiddle
Address Twiddle
twiddleSel
twiddlegen
twidModSel
addr23
addr01
t1BotAccum
t1TwidCnt
cohWrapCnt
t1Cnt
mem_dataInRAM3
mem_dataInRAM2
mem_dataInRAM1
mem_dataInRAM0
dataOutRAM3
dataOutRAM2
dataOutRAM1
dataOutRAM0
wrRAM3
wrRAM2
wrRAM1
wrRAM0
addrRAM3
addrRAM2
addrRAM1
addrRAM0
t1Addr[13:0]
t1Twid[3:0]
wrapCnt[7:0]
t1Cnt[7:0]
dataInRAM3[15:0]
dataInRAM2[15:0]
dataInRAM1[15:0]
dataInRAM0[15:0]
ss2_cohOut3[15:0]
ss2_cohOut2[15:0]
ss2_cohOut1[15:0]
ss2_cohOut0[15:0]
ss2_wrCohRAM3
ss2_wrCohRAM2
ss2_wrCohRAM1
ss2_wrCohRAM0
ss2_addr23[7:0]
ss2_addr01[7:0]
ss2_T1Done
corrLineSize[7:0]
ss2_t1AddrSav[13:0]
ss2_cohBase[13:0]
ss2_twidModSel[1:0]
ss2_t1TwidSav[3:0]
ss2_wrapCntSav[7:0]
ss2_t1Size[7:0]
ss2_t1CntSav[7:0]
mem_ackCohAcc
ss2_reqCohRAM
from/to cohaccmod
from ss2controlmod
from/to cohaccmod

FIG. 41



Twiddle Mux Implementation

FIG. 42

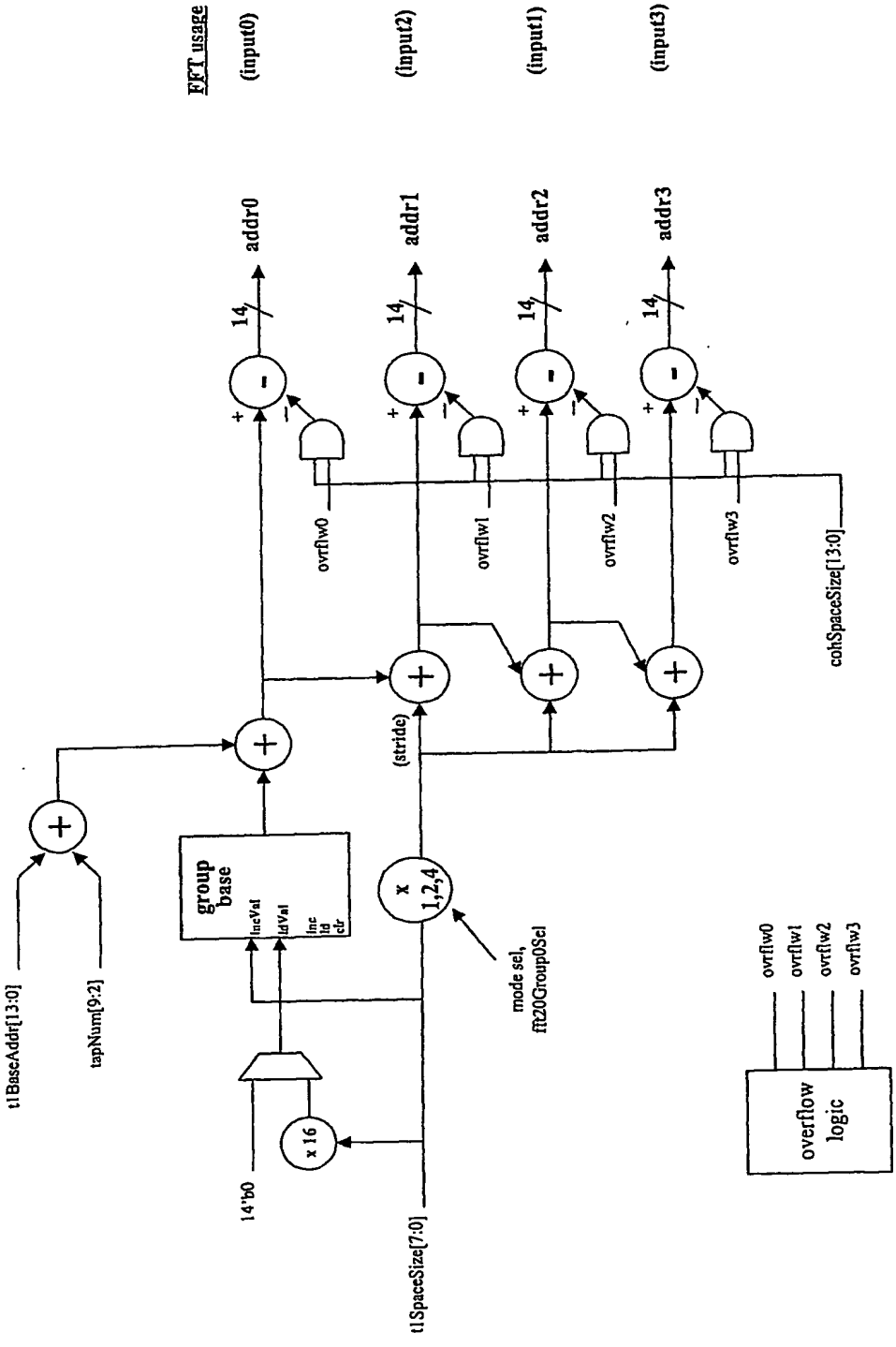


FIG. 43

FIFO2 Output Address Generation

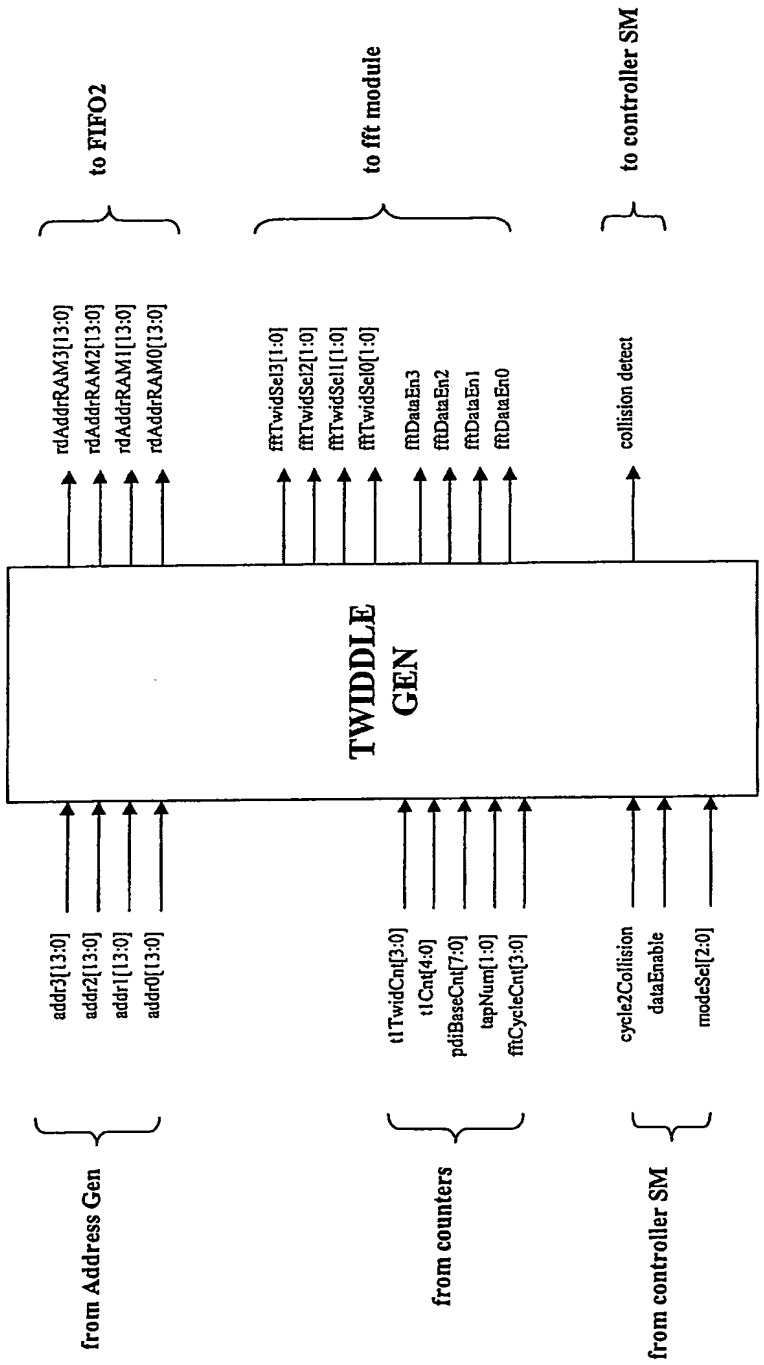


FIG. 44

FFT Address Twiddling

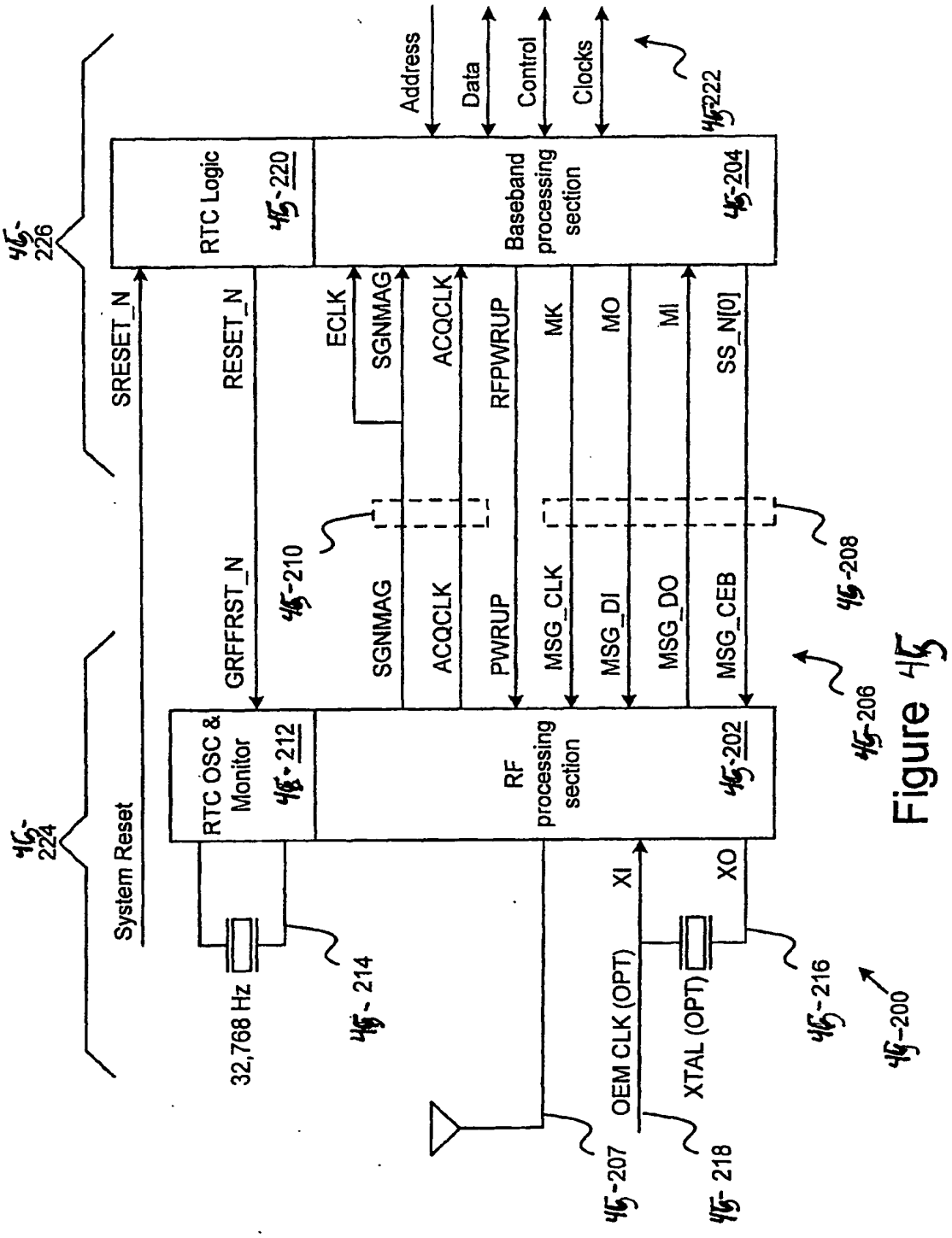

45-226
RTC Logic
45-220
Baseband processing section
45-204
Address
Data
Control
Clocks
45-222
SRESET_N
RESET_N
ECLK
SGNMAG
ACQCLK
RFPWRUP
MK
MO
MI
SS_N[0]
45-210
SGNMAG
ACQCLK
PWRUP
MSG_CLK
MSG_DI
MSG_DO
MSG_CEB
45-208
45-206
GRFFRST_N
45-224
System Reset
RTC OSC & Monitor
45-212
RF processing section
45-202
XI
XO
32,768 Hz
45-214
OEM CLK (OPT)
XTAL (OPT)
45-216
45-200
45-207
45-218

Figure 45

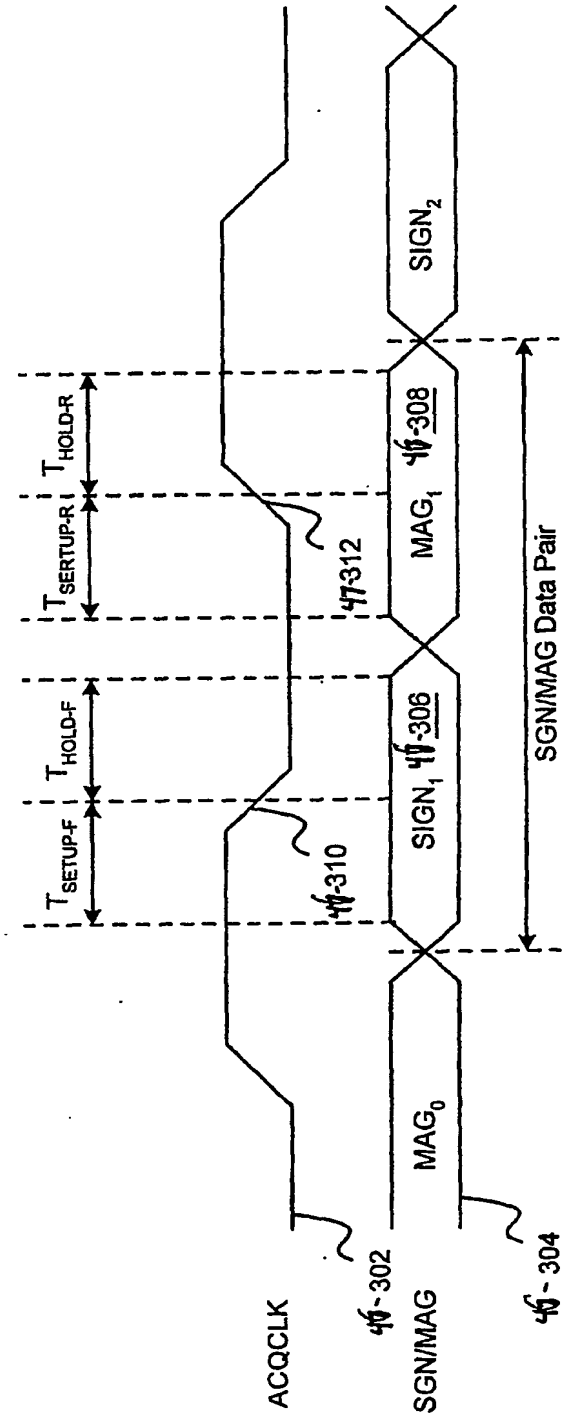

ACQCLK
SGN/MAG
MAG0
SIGN1
MAG1
SIGN2
SGN/MAG Data Pair
TSETUP-F
THOLD-F
TSETUP-R
THOLD-R
310
312
306
308
302
304
300

FIGURE 16

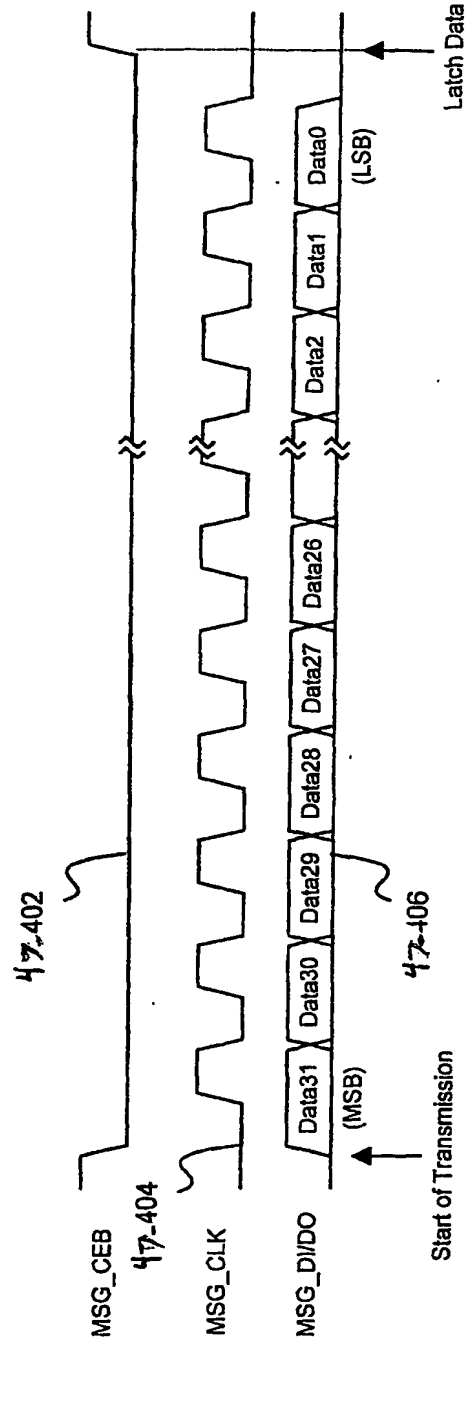

47-402
47-404
MSG_CEB
MSG_CLK
MSG_DI/DO
Data31
Data30
Data29
Data28
Data27
Data26
Data2
Data1
Data0
(MSB)
(LSB)
Start of Transmission
Latch Data
47-406
47-400

FIGURE 47

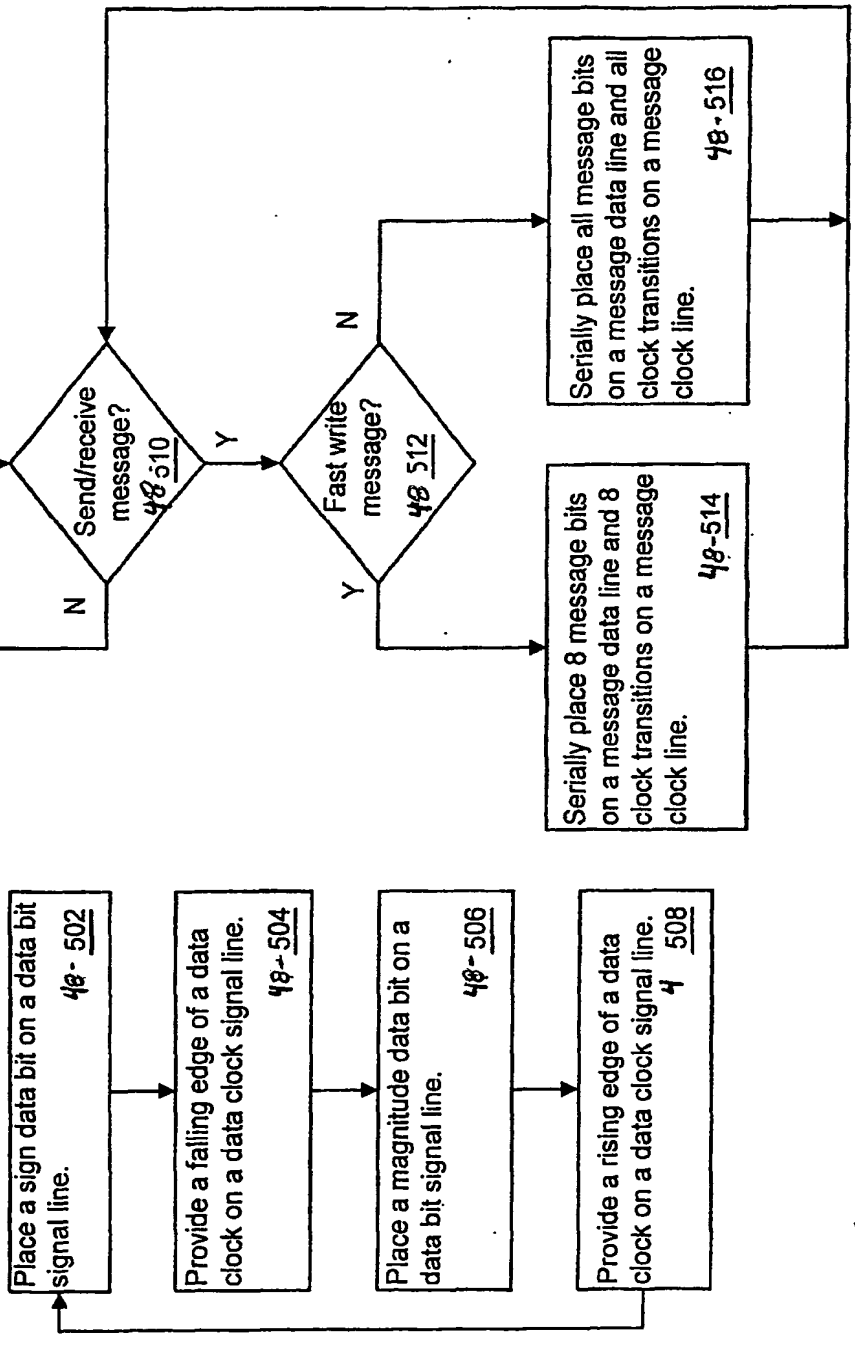

Place a sign data bit on a data bit signal line. 48-502
Provide a falling edge of a data clock on a data clock signal line. 48-504
Place a magnitude data bit on a data bit signal line. 48-506
Provide a rising edge of a data clock on a data clock signal line. 48-508
Send/receive message? 48-510
N
Y
Fast write message? 48-512
N
Y
Serially place 8 message bits on a message data line and 8 clock transitions on a message clock line. 48-514
Serially place all message bits on a message data line and all clock transitions on a message clock line. 48-516
48-500

Figure 48

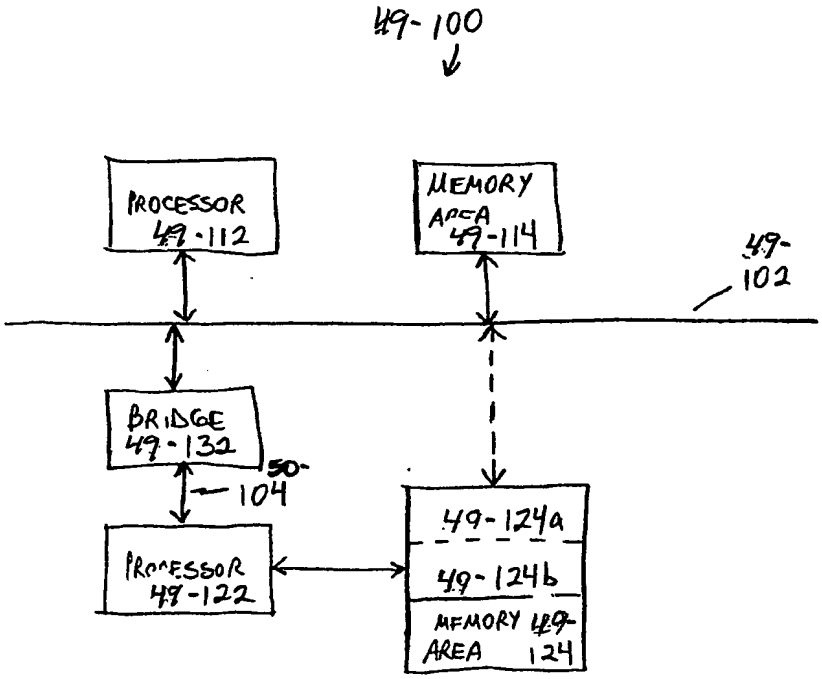
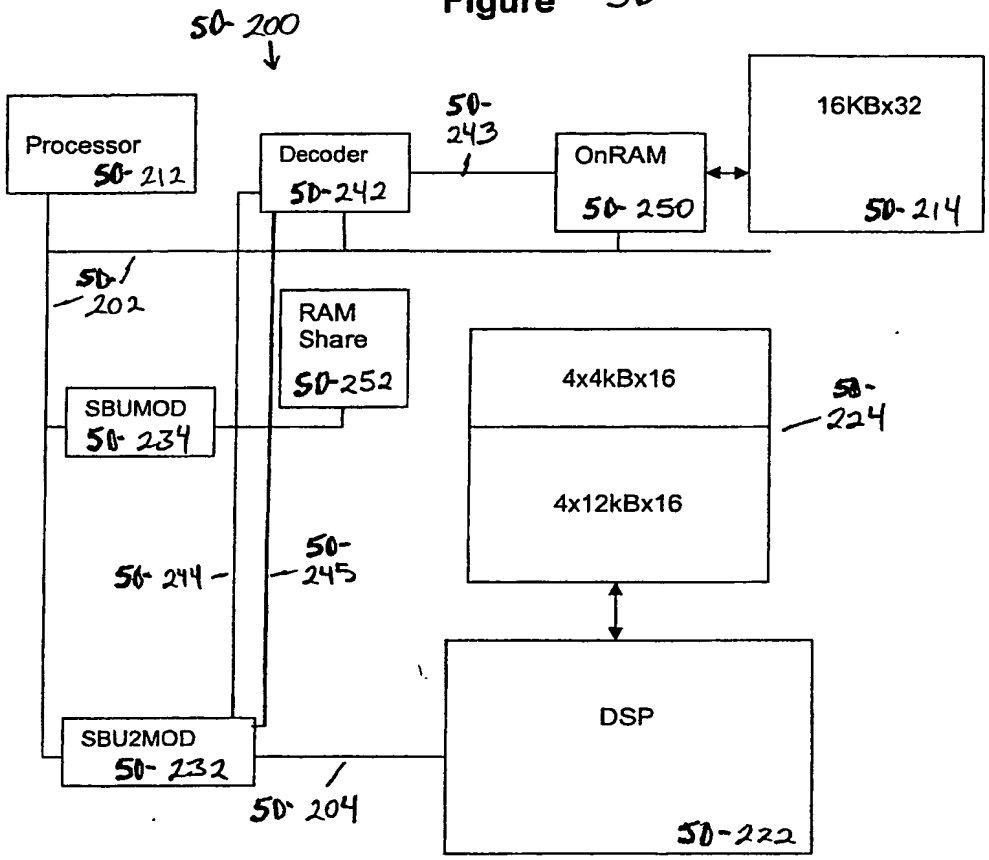

49-100
PROCESSOR 49-112
MEMORY AREA 49-114
49-102
BRIDGE 49-132
50-104
PROCESSOR 49-122
49-124a
49-124b
MEMORY AREA 49-124

FIGURE 49

Figure 5D


50-200
Processor
50-212
Decoder
50-242
50-243
OnRAM
50-250
16KBx32
50-214
50-202
RAM Share
50-252
SBUMOD
50-234
4x4kBx16
4x12kBx16
50-224
50-244
50-245
SBU2MOD
50-232
50-204
DSP
50-222

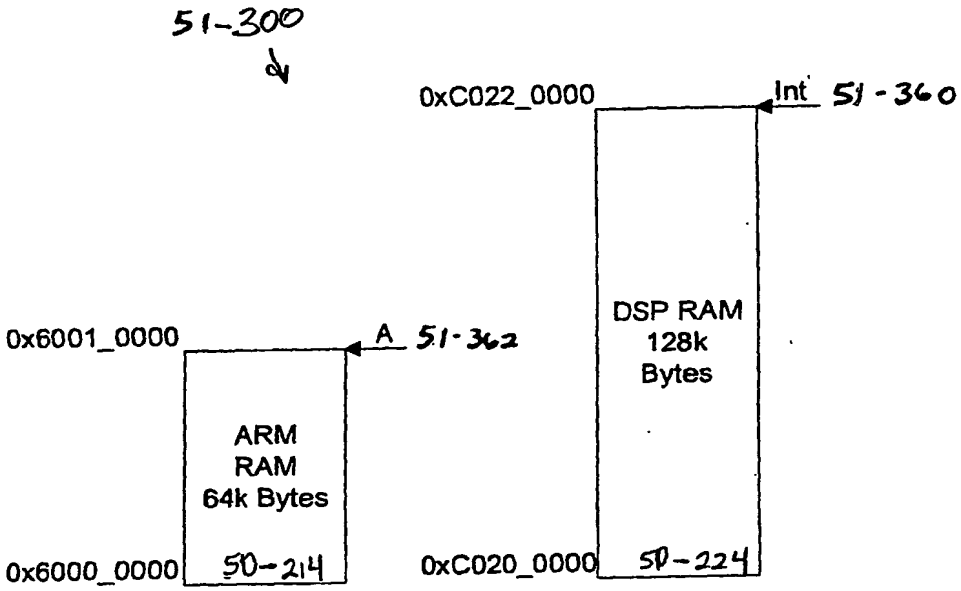
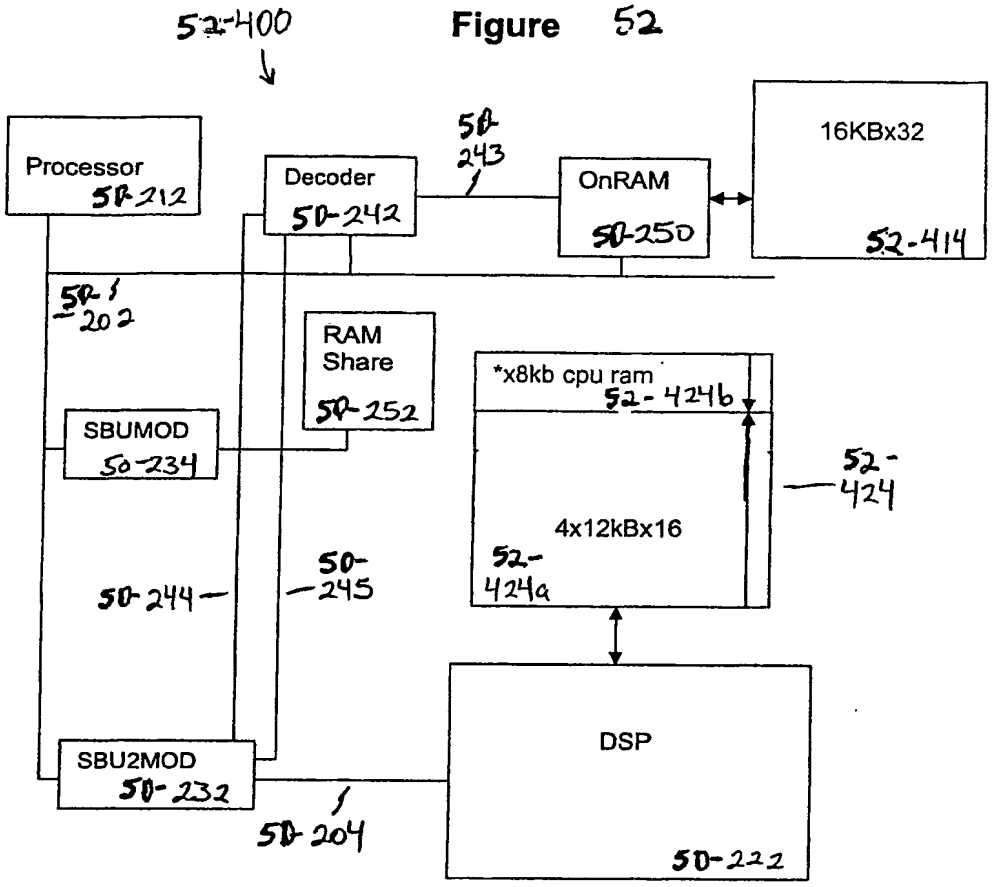

51-300
0xC022_0000
Int 51-360
0x6001_0000
A 51-362
DSP RAM
128k
Bytes
ARM
RAM
64k Bytes
0x6000_0000
50-214
0xC020_0000
50-224

FIGURE 5.1

Figure 52


52-400
Processor
50-212
Decoder
50-242
50-243
OnRAM
50-250
16KBx32
52-414
50-202
RAM Share
50-252
*x8kb cpu ram
52-4246
SBUMOD
50-234
52-424
4x12kBx16
52-424a
50-244
50-245
DSP
SBU2MOD
50-232
50-204
50-222

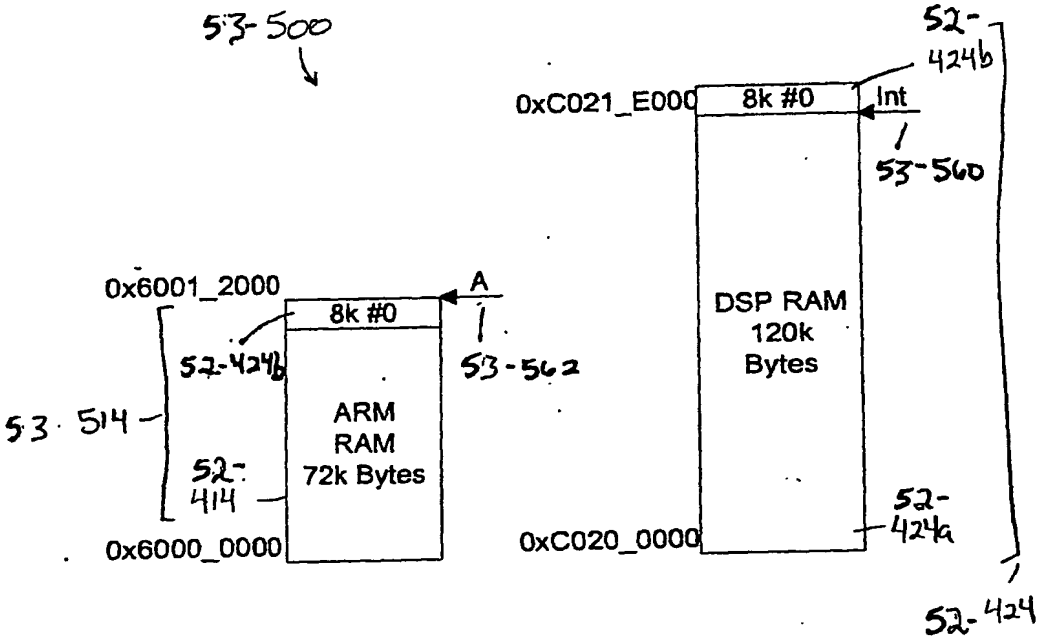

53-500
52-424b
0xC021_E000
8k #0
Int
53-560
0x6001_2000
A
8k #0
52-424b
53-562
53-514
ARM
RAM
72k Bytes
52-414
0x6000_0000
DSP RAM
120k
Bytes
52-424a
0xC020_0000
52-424

FIGURE 53A

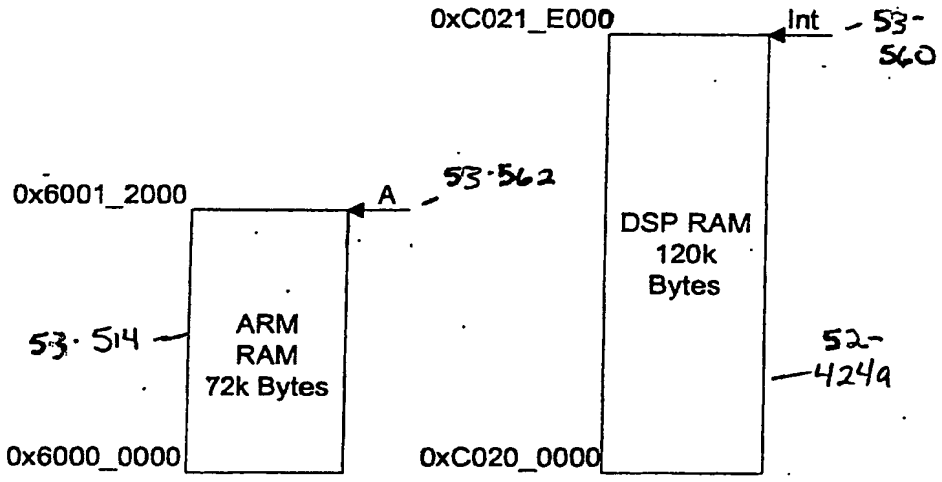

0xC021_E000
Int
53-560
0x6001_2000
A
53-562
53-514
ARM
RAM
72k Bytes
0x6000_0000
DSP RAM
120k
Bytes
52-424a
0xC020_0000

FIGURE 53B

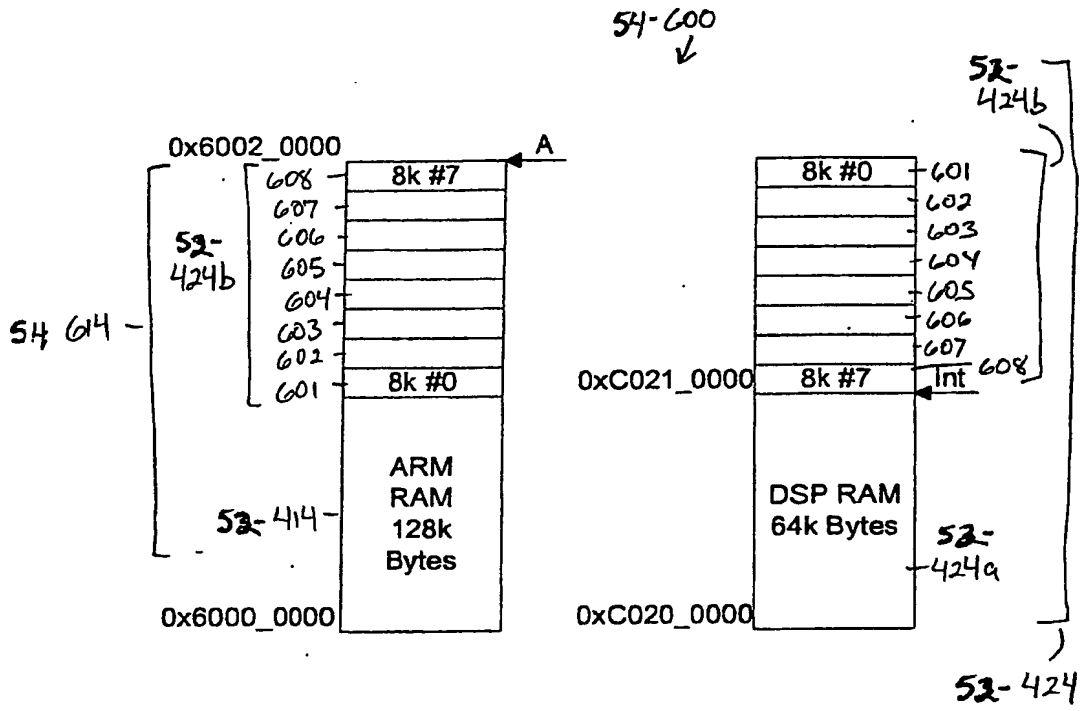

54-600
52-424b
0x6002_0000
A
608
607
606
605
604
603
602
601
8k #7
8k #0
52-424b
54 614
52-414
ARM RAM 128k Bytes
0x6000_0000
8k #0
601
602
603
604
605
606
607
0xC021_0000
8k #7
Int
608
DSP RAM 64k Bytes
52-424a
0xC020_0000
52-424

FIGURE 54A

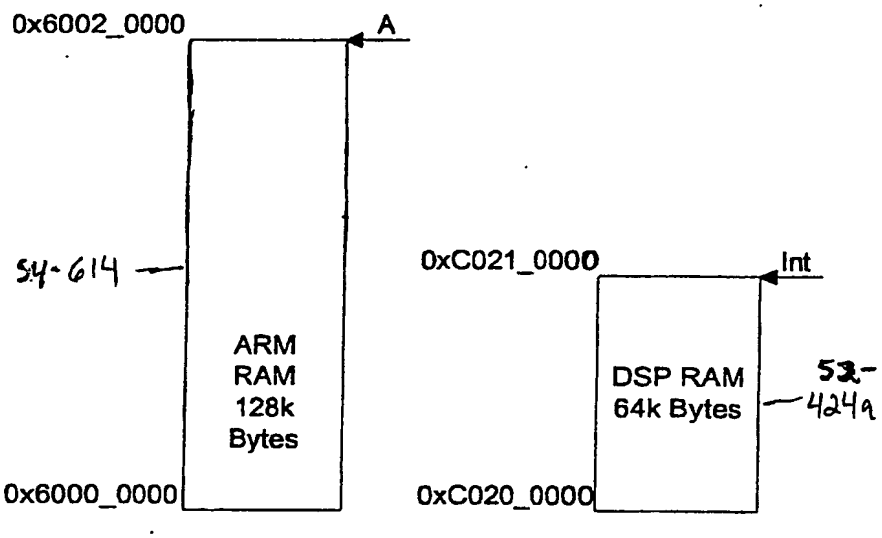

0x6002_0000
A
54-614
ARM
RAM
128k
Bytes
0x6000_0000
0xC021_0000
Int
DSP RAM
64k Bytes
52-
424a
0xC020_0000

FIGURE 54B

Figure 55

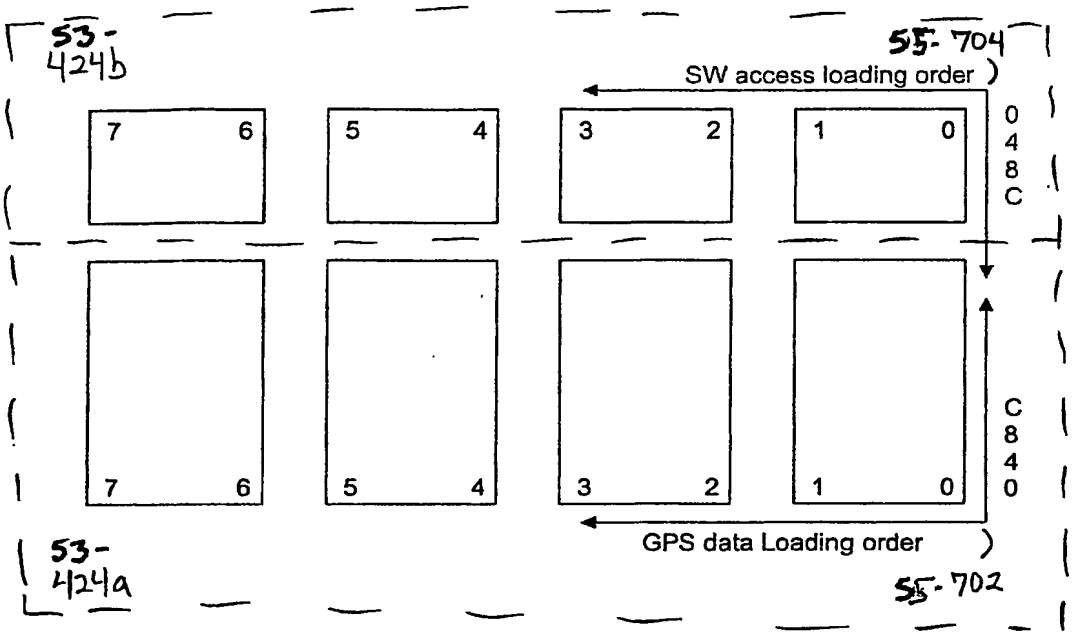
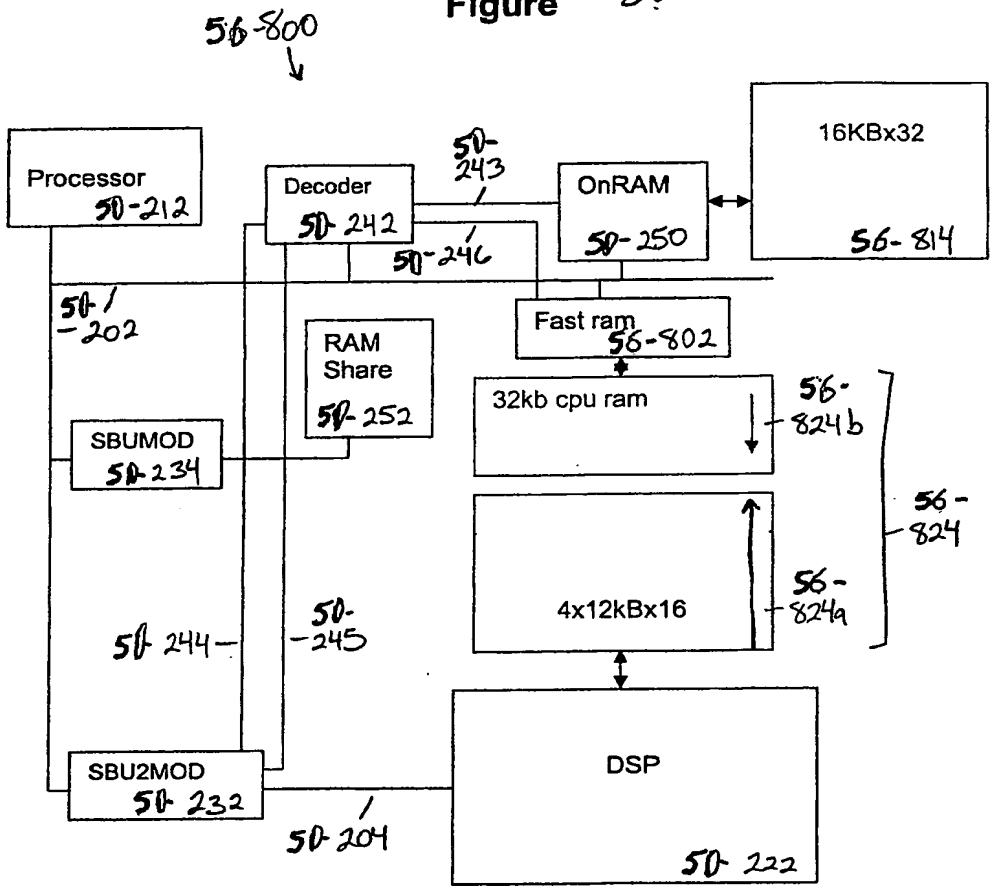

53-424b
55-704
SW access loading order
7 6
5 4
3 2
1 0
0 4 8 C
C 8 4 0
7 6
5 4
3 2
1 0
GPS data Loading order
53-424a
55-702

Figure 56


56-800
Processor
50-212
Decoder
50-242
50-243
OnRAM
50-250
16KBx32
56-814
50-246
50-202
RAM
Share
50-252
Fast ram
56-802
32kb cpu ram
56-824b
SBUMOD
50-234
56-824
4x12kBx16
56-824a
50-244
50-245
SBU2MOD
50-232
DSP
50-222
50-204

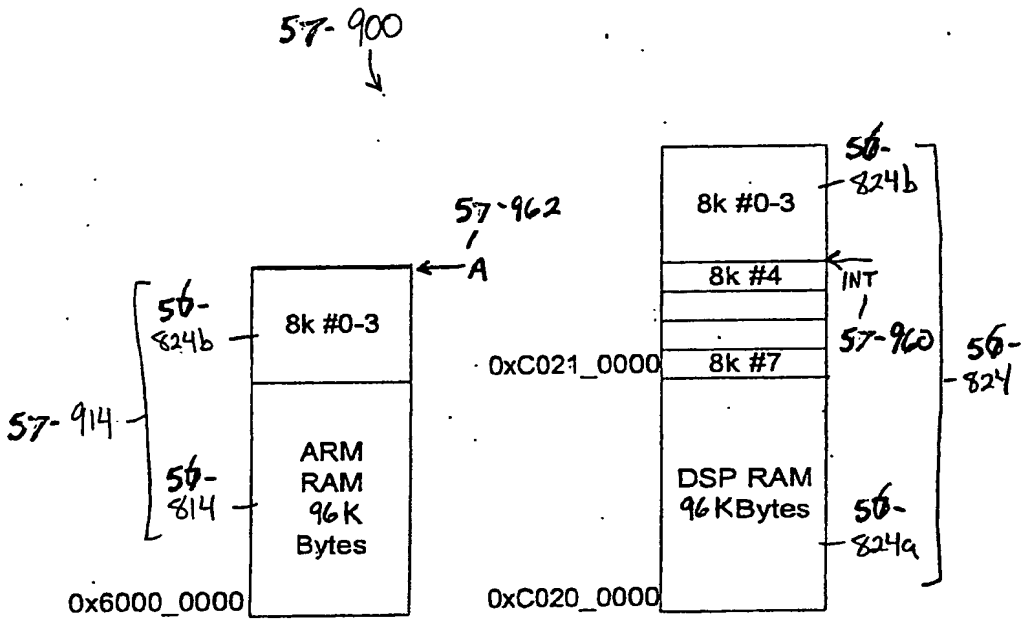

57-900
57-962
A
56-824b
8k #0-3
57-914
56-814
ARM RAM 96K Bytes
0x6000_0000
8k #0-3
8k #4
INT
57-960
0xC021_0000
8k #7
DSP RAM 96KBytes
0xC020_0000
56-824b
56-824
56-824a

FIGURE 57A

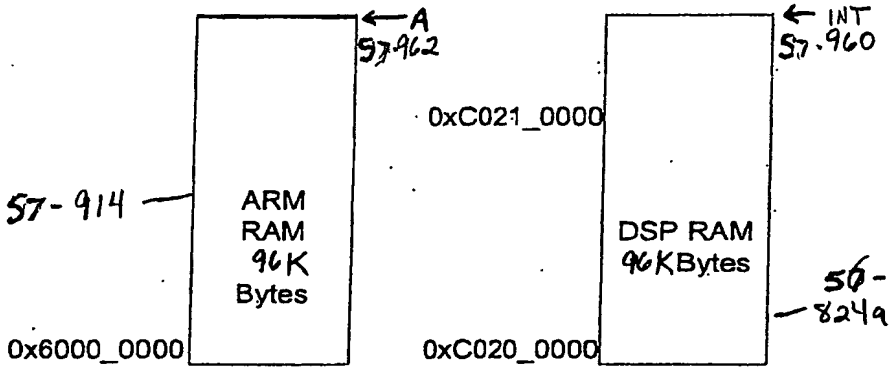
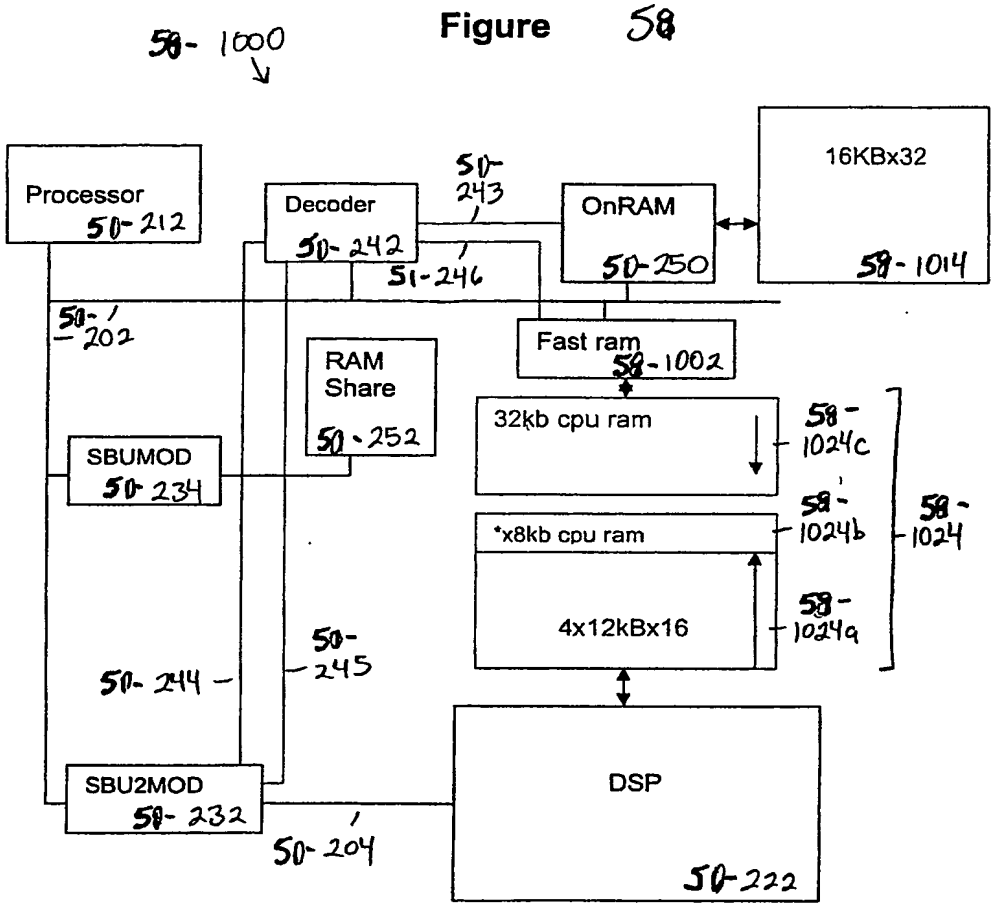

A
57-962
INT
57-960
0xC021_0000
57-914
ARM
RAM
96K
Bytes
DSP RAM
96KBytes
50-
824a
0x6000_0000
0xC020_0000

FIGURE 57.3

Figure 58


50-1000
Processor
50-212
Decoder
50-242
50-243
OnRAM
50-250
16KBx32
58-1014
51-246
50-202
RAM Share
50-252
Fast ram
58-1002
32kb cpu ram
58-1024c
SBUMOD
50-234
*x8kb cpu ram
58-1024b
58-1024
4x12kBx16
58-1024a
50-244
50-245
DSP
50-222
SBU2MOD
50-232
50-204

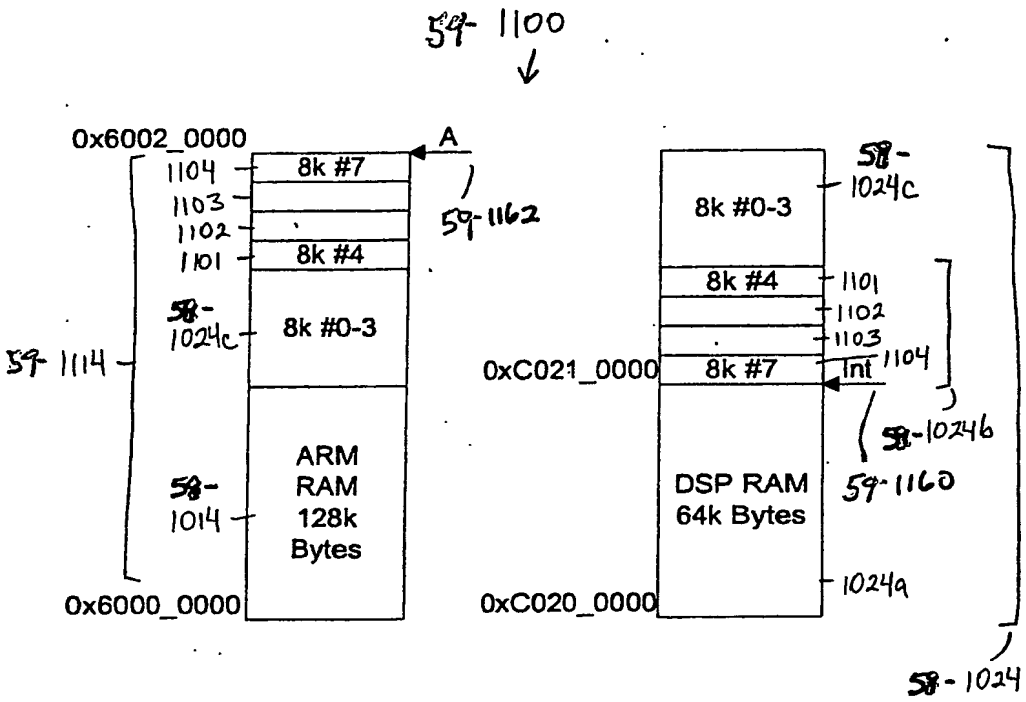

59- 1100
0x6002_0000
A
1104
8k #7
1103
1102
1101
8k #4
58-
1024c
8k #0-3
59- 1114
ARM
RAM
128k
Bytes
58-
1014
0x6000_0000
59- 1162
8k #0-3
58-
1024c
8k #4
1101
1102
1103
0xC021_0000
8k #7
Int
1104
58-1024b
DSP RAM
64k Bytes
59-1160
1024a
0xC020_0000
58-1024

FIGURE 59A

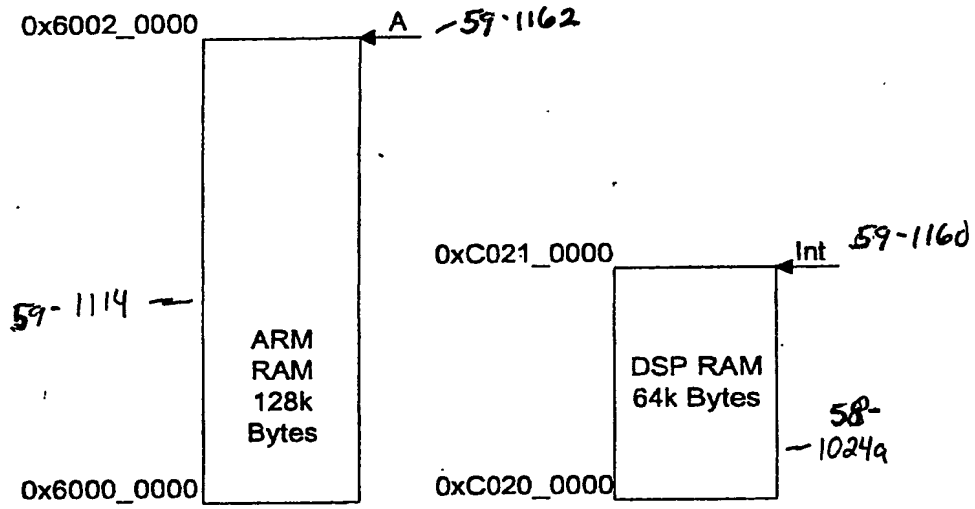

0x6002_0000
A
59-1162
59-1114
ARM
RAM
128k
Bytes
0x6000_0000
0xC021_0000
Int
59-1160
DSP RAM
64k Bytes
58-
1024a
0xC020_0000

FIGURE 59B

Figure 60

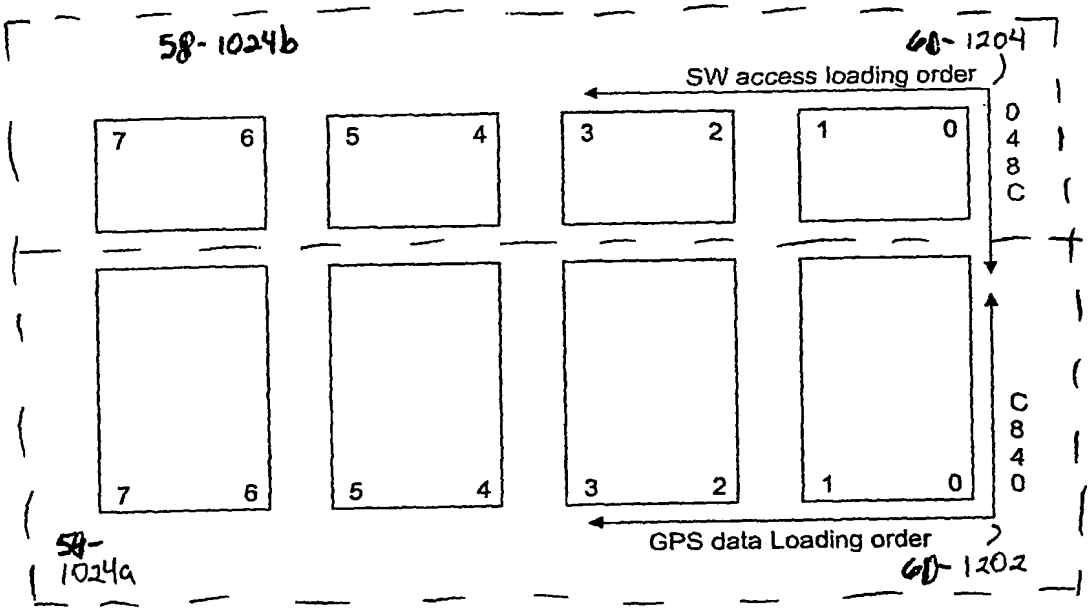
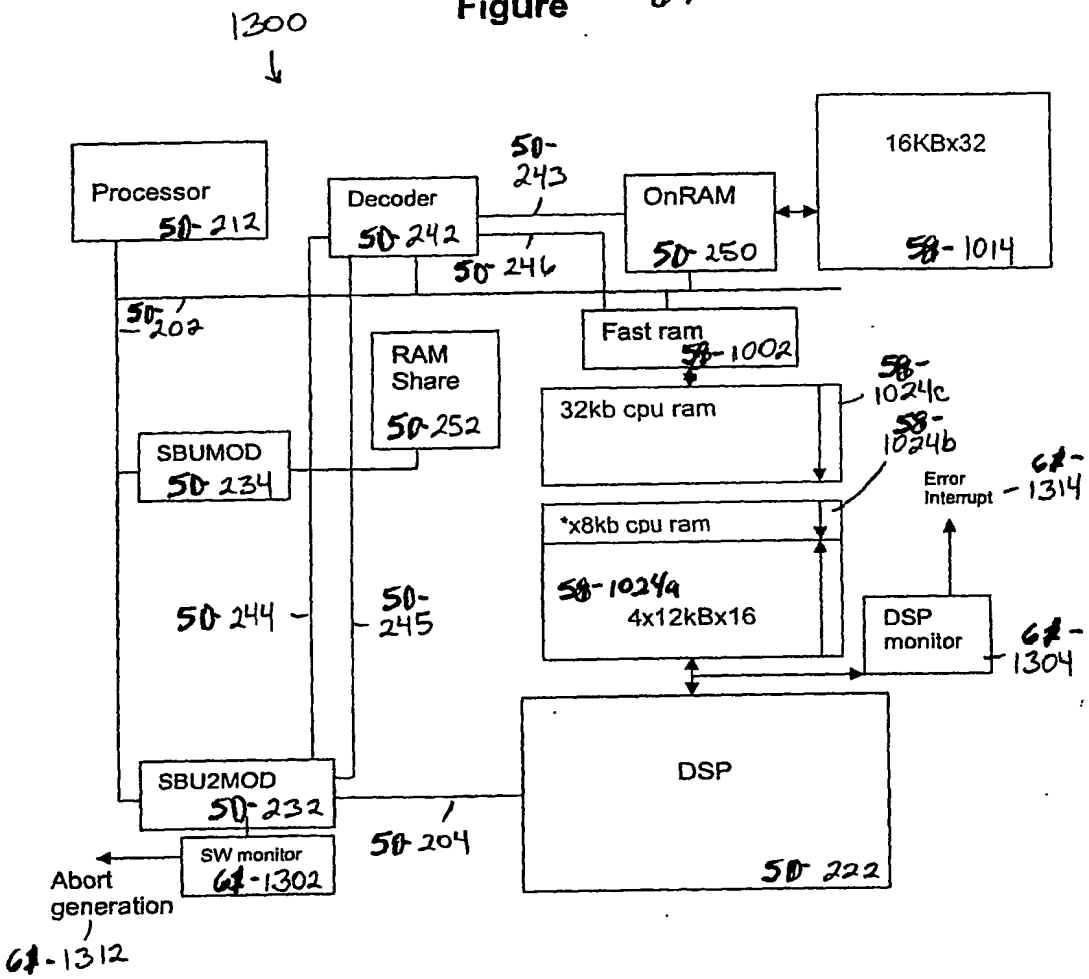

58-1024b
60-1204
SW access loading order
7 6
5 4
3 2
1 0
0 4 8 C
C 8 4 0
7 6
5 4
3 2
1 0
GPS data Loading order
58-1024a
60-1202

Figure 61


1300
Processor
50-212
Decoder
50-242
50-243
50-246
OnRAM
50-250
16KBx32
58-1014
50-202
Fast ram
58-1002
RAM Share
50-252
32kb cpu ram
58-1024c
58-1024b
SBUMOD
50-234
Error Interrupt
61-1314
*x8kb cpu ram
58-1024a
4x12kBx16
50-244
50-245
DSP monitor
61-1304
DSP
50-222
SBU2MOD
50-232
50-204
SW monitor
61-1302
Abort generation
61-1312

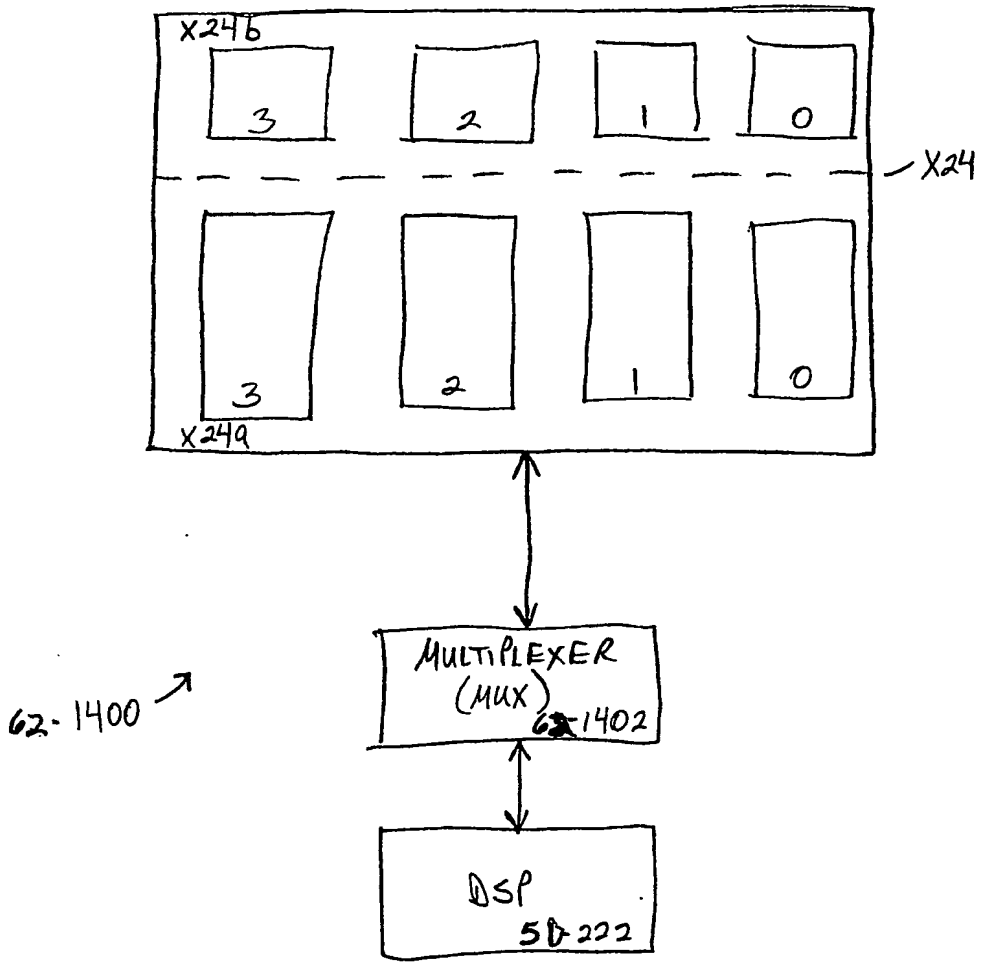


FIGURE 62

RAM_CTL: (address = 0xC0000000)

63- 1500

	R	R	R	R	R	R	R	RW
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-		-	

	RW	RW	RW	RW	RW	RW	RW	RW
Reset value	0	0	0	0	0	0	0	0
		SWI_ENB	MAP_BLK[2]	MAP_BLK[1]	MAP_BLK[0]	DSP64K_ MAP_ENB	EN_CPU_WAB	EN_CPU_RAB

Figure 63

RAM_STA: (address = 0xC0000004) 64-1600

	15	14	13	12	11	10	9	8
	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-

	7	6	5	4	3	2	1	0
	R	R	R	R	R	R	RW	RW
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	CPUW_VIO	CPUR_VIO

Figure 64

DSP_ADDR: (address = 0xC0000008)

65-1700

	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	DSP_ADD [15]	•	•	•	•	•	•	•

	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	•	•	•	•	•	•	•	DSP_ADD [0]

Figure 65

Figure 6A 66.1800

Number of 8Kbyte Blocks Mapped	DSP32K_SWI_ENB	DSP64K_MAP_ENB	MAP_BLK[2:0]	DSP Address Range	CPU Normal Mapped SBU2 Address Range	CPU Soft Mapped SBU2 Address Range	CPU Hard Switch ASB Address Range
1	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	000	0x0000_0000–0x0001_DFFF	0xC021_E000–0xC021_FFFF	0x6001_0000–0x6001_1FFF	NA
2	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	001	0x0000_0000–0x0001_BFFF	0xC021_C000–0xC021_FFFF	0x6001_0000–0x6001_3FFF	NA
3	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	010	0x0000_0000–0x0001_9FFF	0xC021_A000–0xC021_FFFF	0x6001_0000–0x6001_5FFF	NA
4	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	011	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	0x6001_0000–0x6001_7FFF	NA
5	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	100	0x0000_0000–0x0001_5FFF	0xC021_6000–0xC021_FFFF	0x6001_8000–0x6001_9FFF	0x6001_6000–0x6001_7FFF
	0	1	100	0x0000_0000–0x0001_5FFF	0xC021_6000–0xC021_FFFF	0x6001_0000–0x6001_9FFF	NA
6	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	101	0x0000_0000–0x0001_3FFF	0xC021_4000–0xC021_FFFF	0x6001_8000–0x6001_BFFF	0x6001_0000–0x6001_7FFF
	0	1	101	0x0000_0000–0x0001_3FFF	0xC021_4000–0xC021_FFFF	0x6001_0000–0x6001_BFFF	NA
7	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA

Number of 8Kbyte Blocks Mapped	DSP32K_SWI_ENB	DSP64K_MAP_ENB	MAP_BLK[2:0]	DSP Address Range	CPU Normal Mapped SBU2 Address Range	CPU Soft Mapped SBU2 Address Range	CPU Hard Switch ASB Address Range
	1	0	XXX	0x0000_0000– 0x0001_7FFF	0xC021_8000– 0xC021_FFFF	NA	0x6001_0000– 0x6001_7FFF
	1	1	110	0x0000_0000– 0x0001_1FFF	0xC021_2000– 0xC021_FFFF	0x6001_8000– 0x6001_DFFF	0x6001_0000– 0x6001_7FFF
	0	1	110	0x0000_0000– 0x0001_1FFF	0xC021_2000– 0xC021_FFFF	0x6001_0000– 0x6001_DFFF	NA
8	0	0	XXX	0x0001_0000– 0x0001_FFFF	0xC020_0000– 0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000– 0x0001_7FFF	0xC021_8000– 0xC021_FFFF	NA	0x6001_0000– 0x6001_7FFF
	1	1	111	0x0000_0000– 0x0000_FFFF	0xC021_0000– 0xC021_FFFF	0x6001_8000– 0x6001_EFFF	0x6001_0000– 0x6001_7FFF
	0	1	111	0x0000_0000– 0x0000_FFFF	0xC021_0000– 0xC021_FFFF	0x6001_0000– 0x6001_FFFF	NA

Figure 66B

66-1800

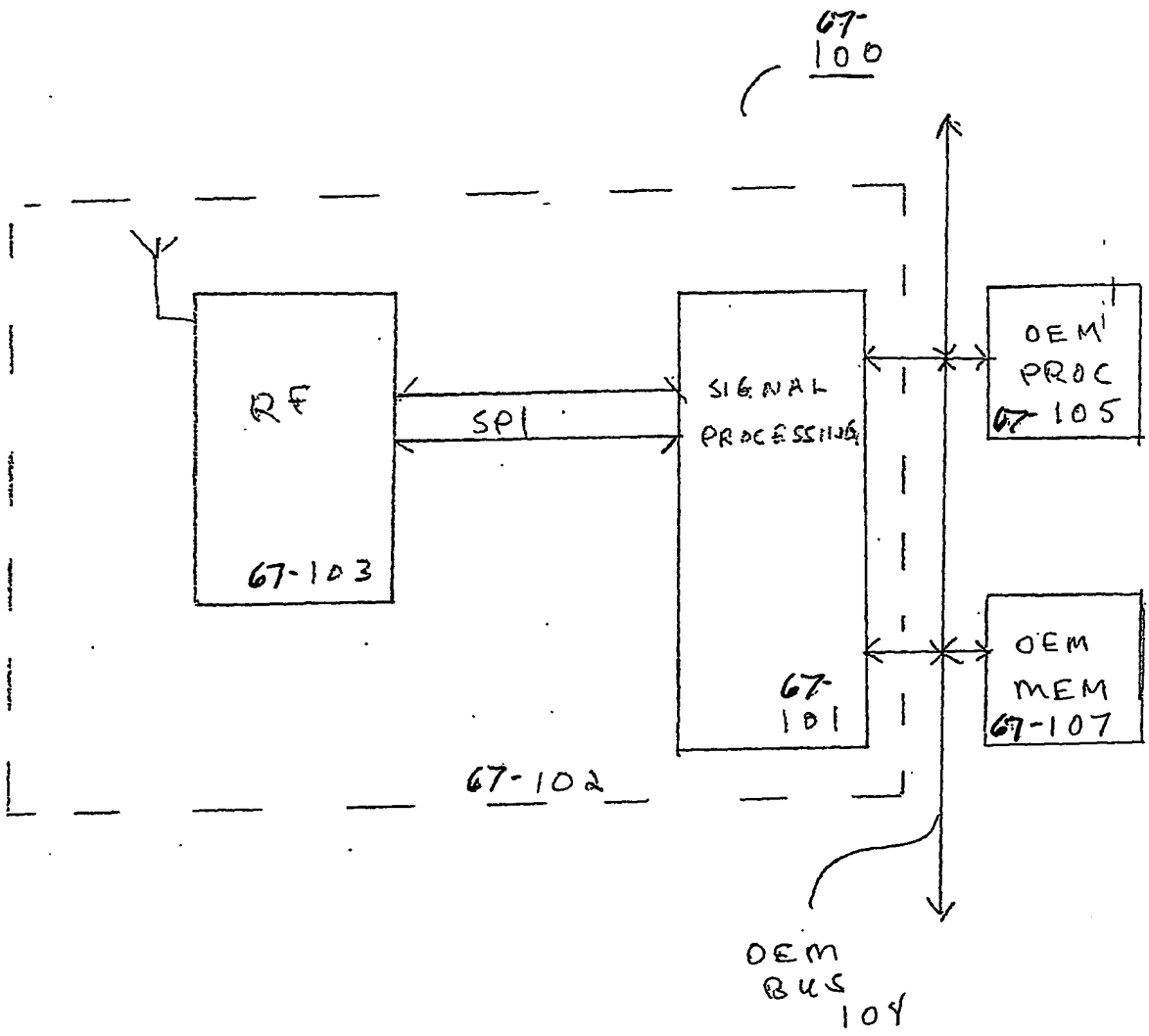

67-100
RF
67-103
SPI
SIGNAL PROCESSING
67-101
67-102
OEM PROC
67-105
OEM MEM
67-107
OEM BUS
104

FIG. 67

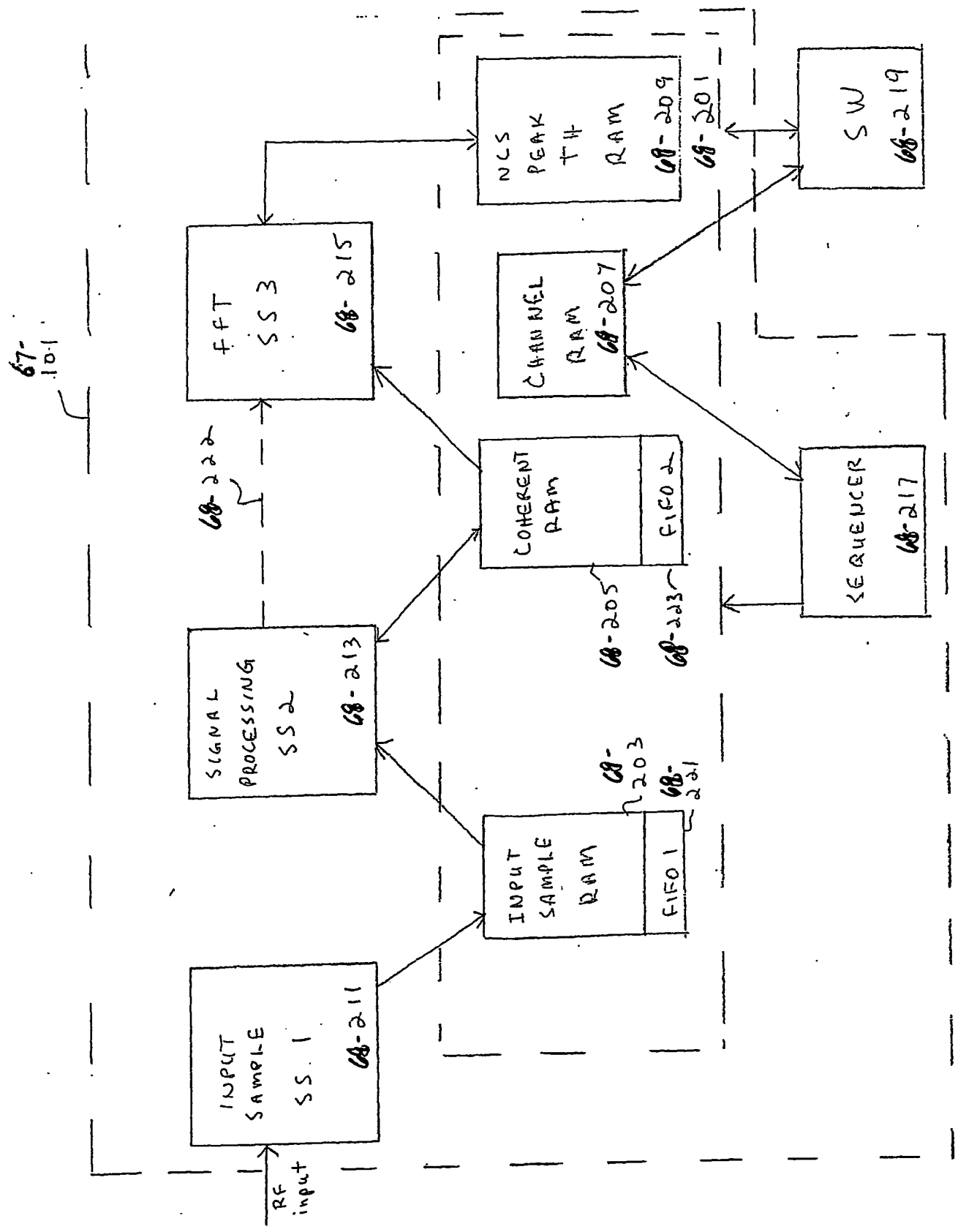

67-101
RF input
INPUT SAMPLE SS 1
68-211
SIGNAL PROCESSING SS 2
68-213
FFT SS 3
68-215
68-222
INPUT SAMPLE RAM
68-203
FIFO 1
68-231
COHERENT RAM
68-205
FIFO 2
68-233
CHANNEL RAM
68-207
NCS PEAK TH RAM
68-209
68-201
SEQUENCER
68-217
SW
68-219

FIG. 6B

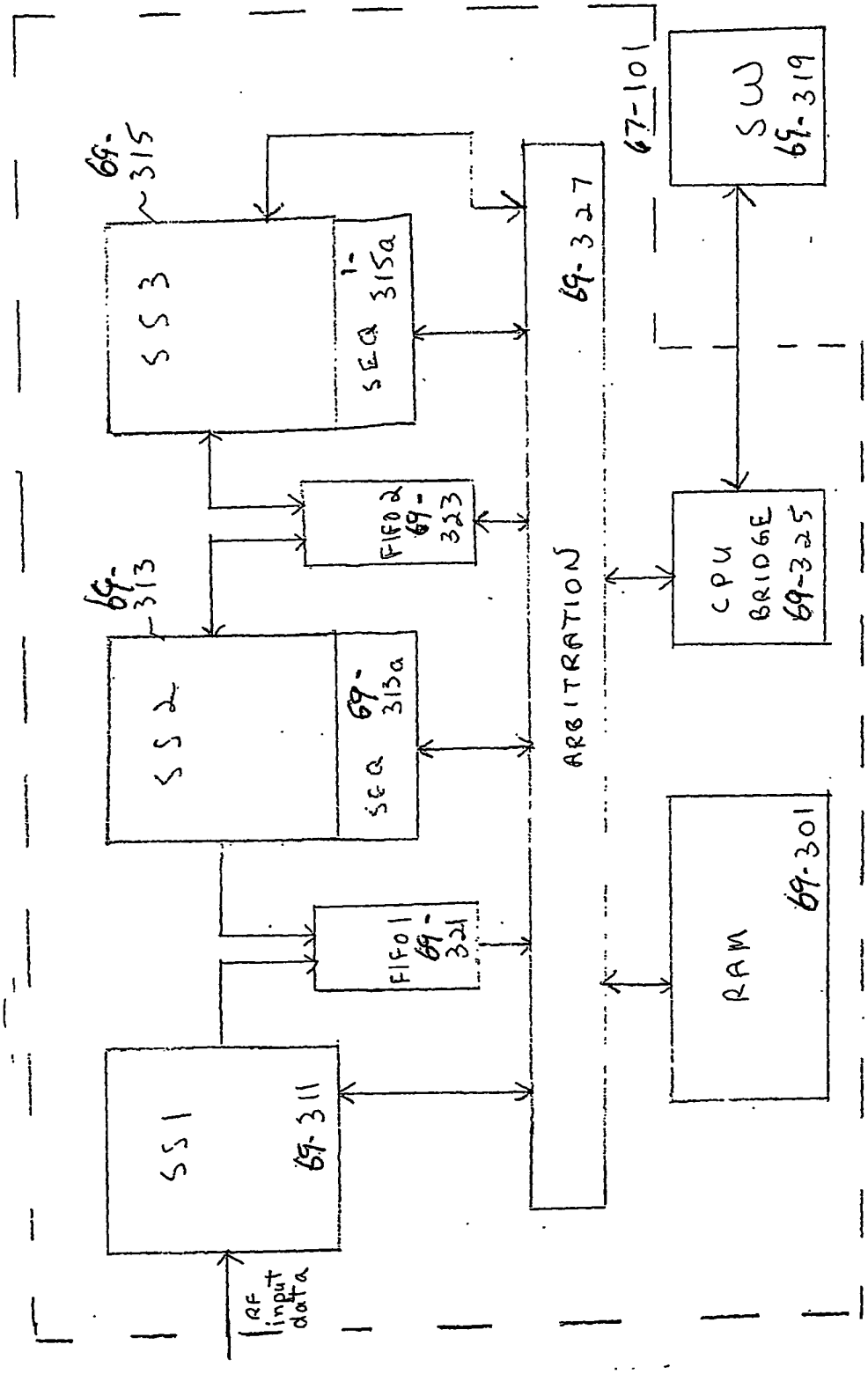

SSS 3
SEQ 315a
69-315
FIFO 2 69-323
SSS 2
SEQ 69-313a
69-313
FIFO 1 69-321
SSS 1
69-311
RF input data
ARBITRATION
69-327
67-101
SW 69-319
CPU BRIDGE 69-325
RAM 69-301

FIG. 69

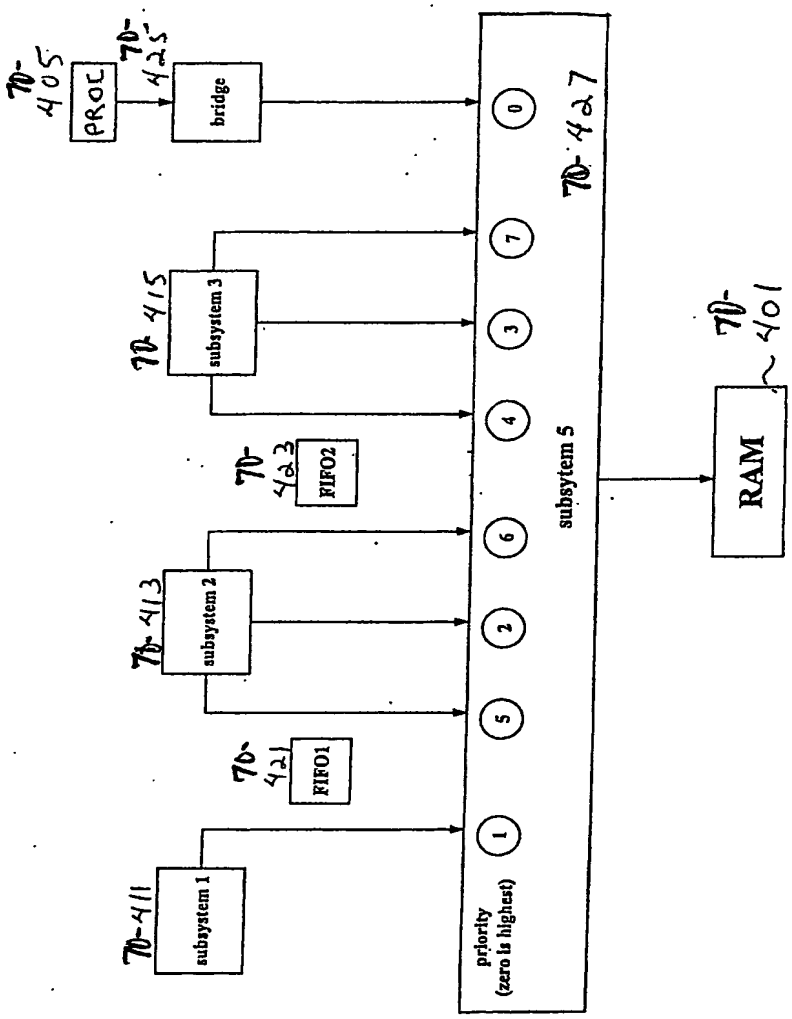

70-405
PROC
70-425
bridge
70-415
subsystem 3
70-423
FIFO2
70-413
subsystem 2
70-421
FIFO1
70-411
subsystem 1
priority
(zero is highest)
0
7
3
4
6
2
5
1
70-427
subsystem 5
70-401
RAM

FIG. 70

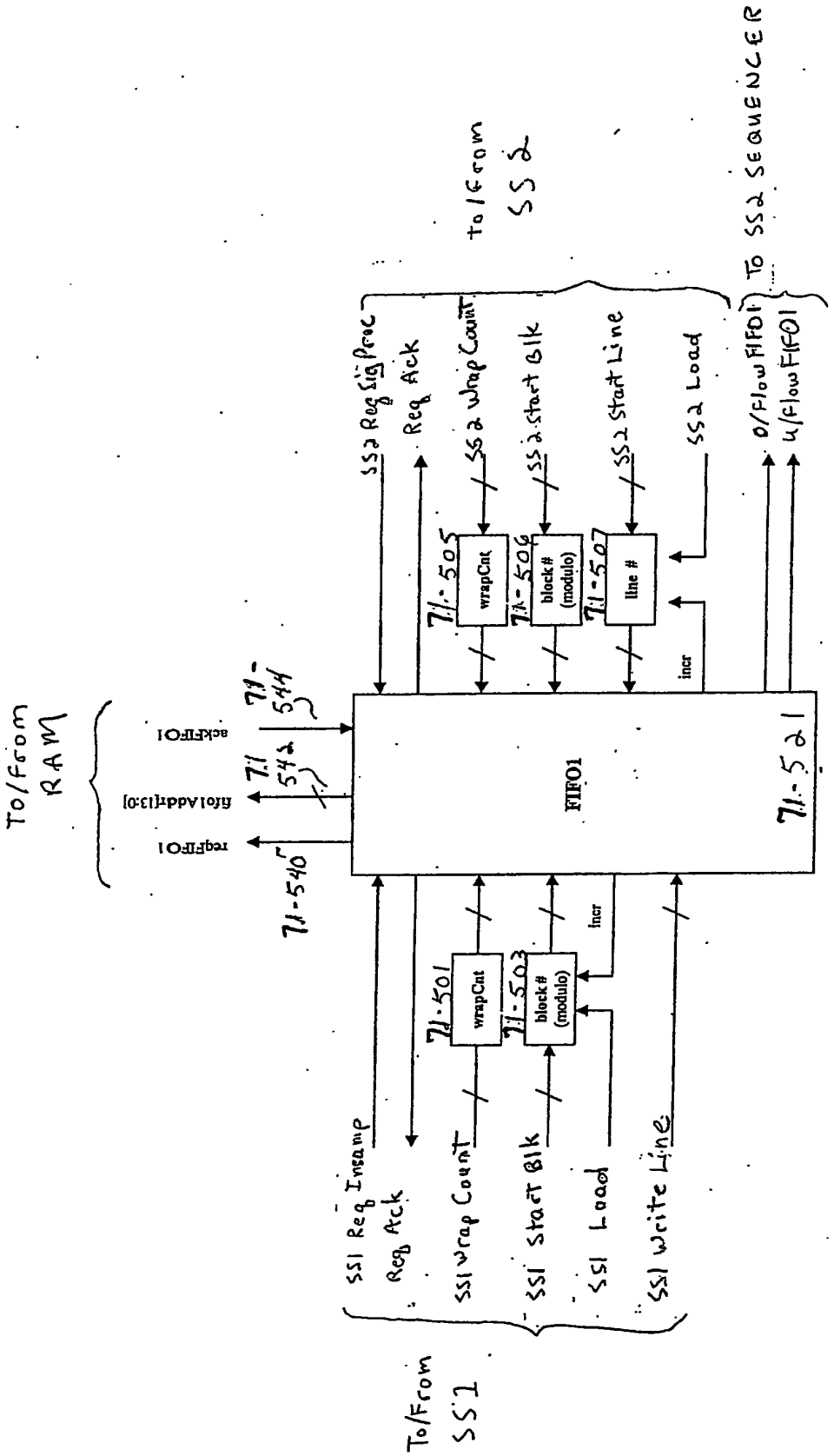

To/From RAM
ackFIFO1
fifo1Addr[13:0]
reqFIFO1
71-544
71-542
71-540
FIFO1
71-541
SS2 Req Sig Proc
Req Ack
SS2 a Wrap Count
SS2 a Start Blk
SS2 Start Line
SS2 Load
O/Flow FIFO1
U/Flow FIFO1
To SS2 SEQUENCER
To/From SS2
wrapCnt
block # (modulo)
line #
incr
71-505
71-506
71-507
SS1 Req Insamp
Req Ack
SS1 Wrap Count
SS1 Start Blk
SS1 Load
SS1 Write Line
To/From SS1
71-501
71-503

FIG. 71.

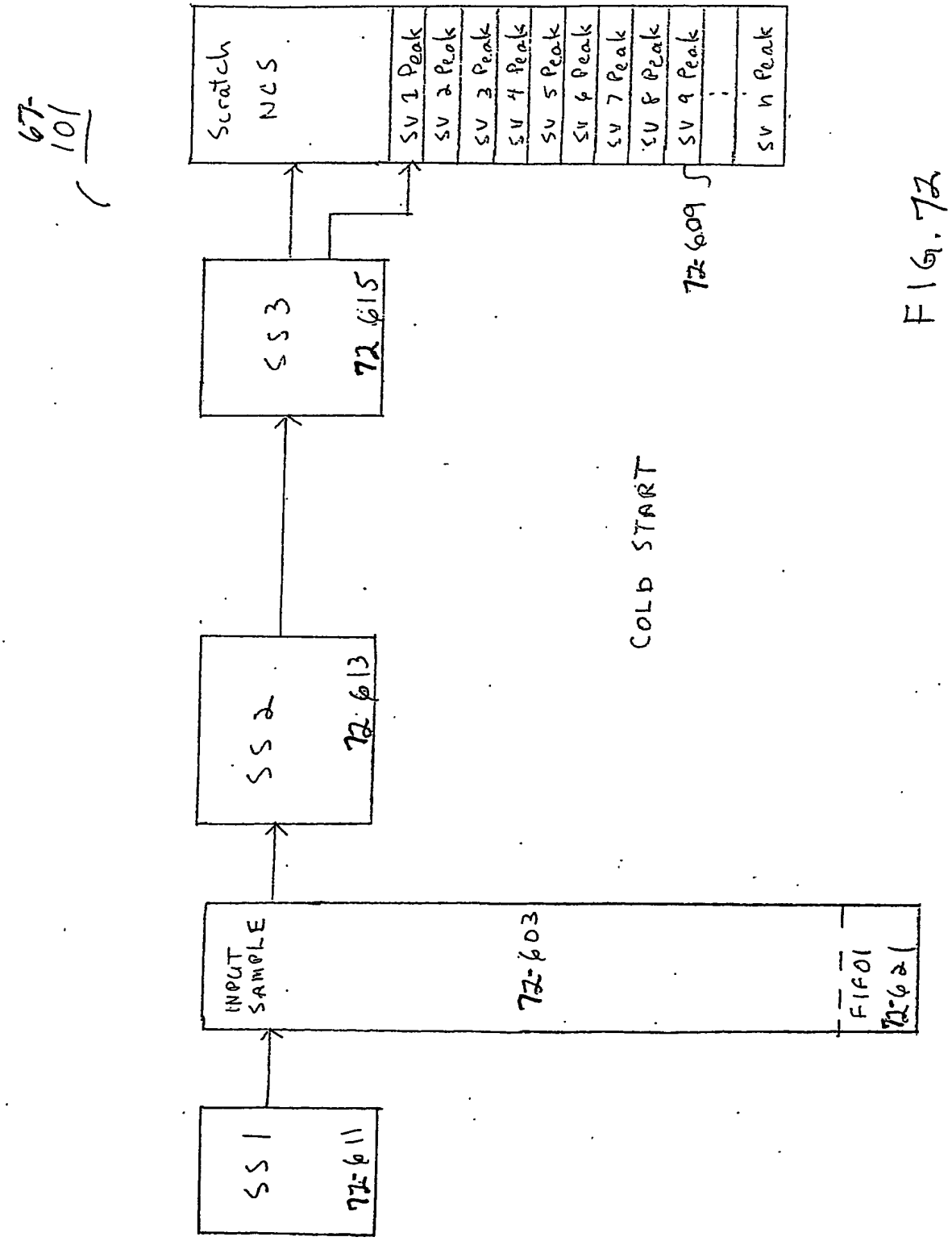

67-101
Scratch NCS
SV 1 Peak
SV 2 Peak
SV 3 Peak
SV 4 Peak
SV 5 Peak
SV 6 Peak
SV 7 Peak
SV 8 Peak
SV 9 Peak
SV n Peak
72-609
SS3
72-615
SS2
72-613
COLD START
INPUT SAMPLE
72-603
FIFO1
72-621
SS1
72-611

FIG. 72

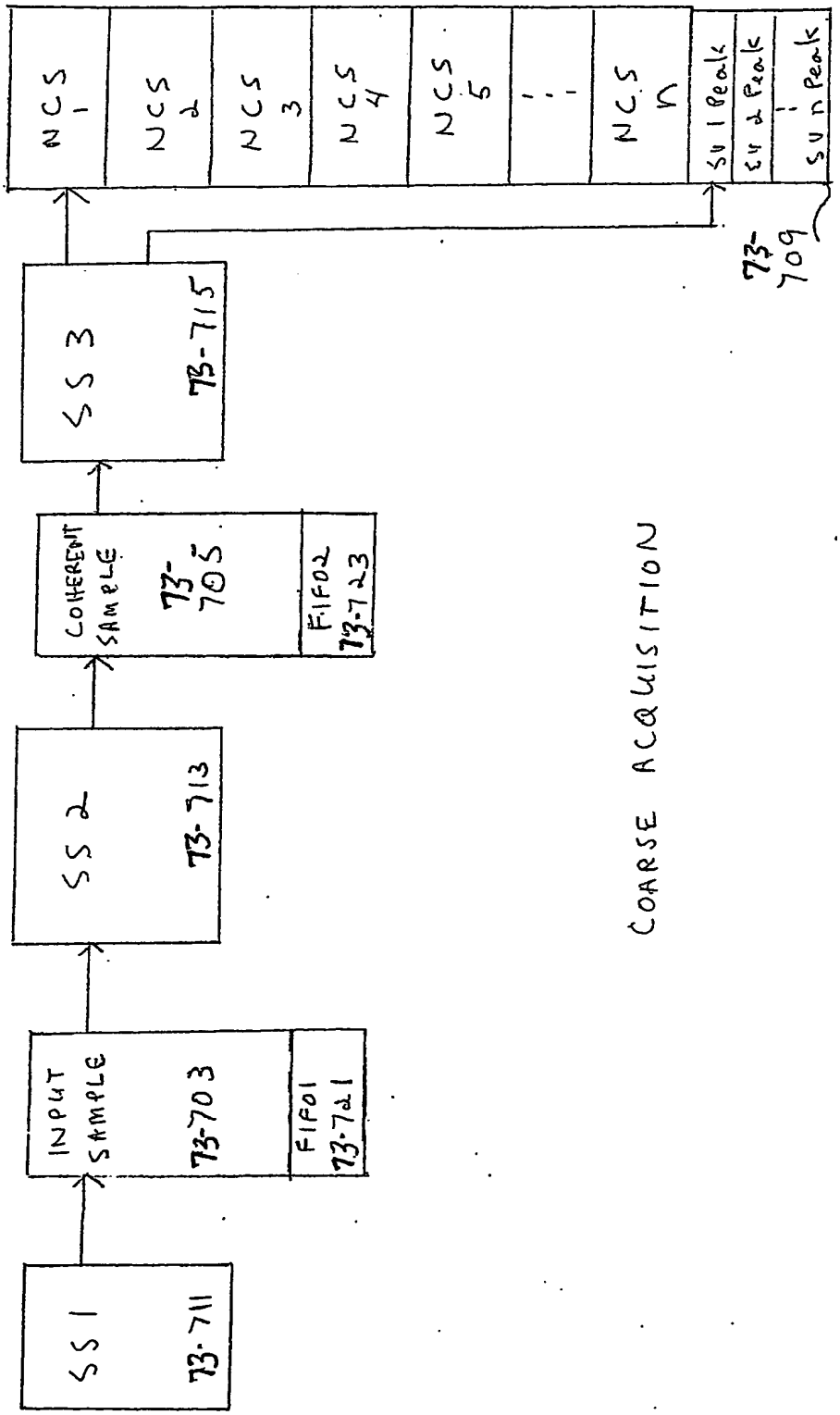

NCS 1
NCS 2
NCS 3
NCS 4
NCS 5
...
NCS N
SV 1 Peak
SV 2 Peak
SV n Peak
73-709
SS3 73-715
COHERENT SAMPLE 73-705
FIFO2 73-723
SS2 73-713
INPUT SAMPLE 73-703
FIFO1 73-721
SS1 73-711
COARSE ACQUISITION

FIG. 73

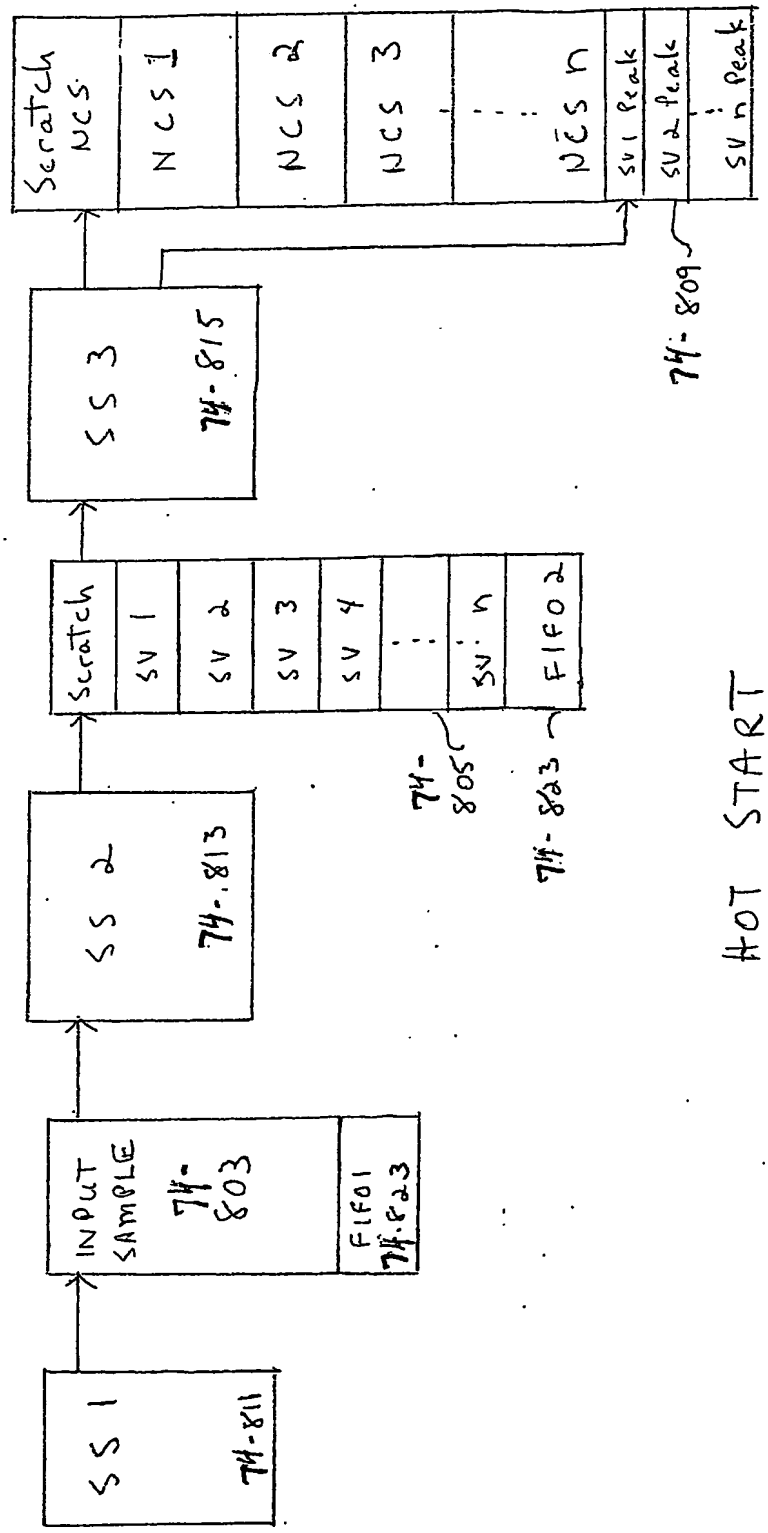

SS 1
74-811
INPUT SAMPLE 74-803
FIFO1 74-823
SS 2
74-813
Scratch
SV 1
SV 2
SV 3
SV 4
SV n
FIFO2
74-805
74-823
SS 3
74-815
Scratch NCS
NCS 1
NCS 2
NCS 3
NCS n
SV 1 Peak
SV 2 Peak
SV n Peak
74-809
HOT START

FIG. 74

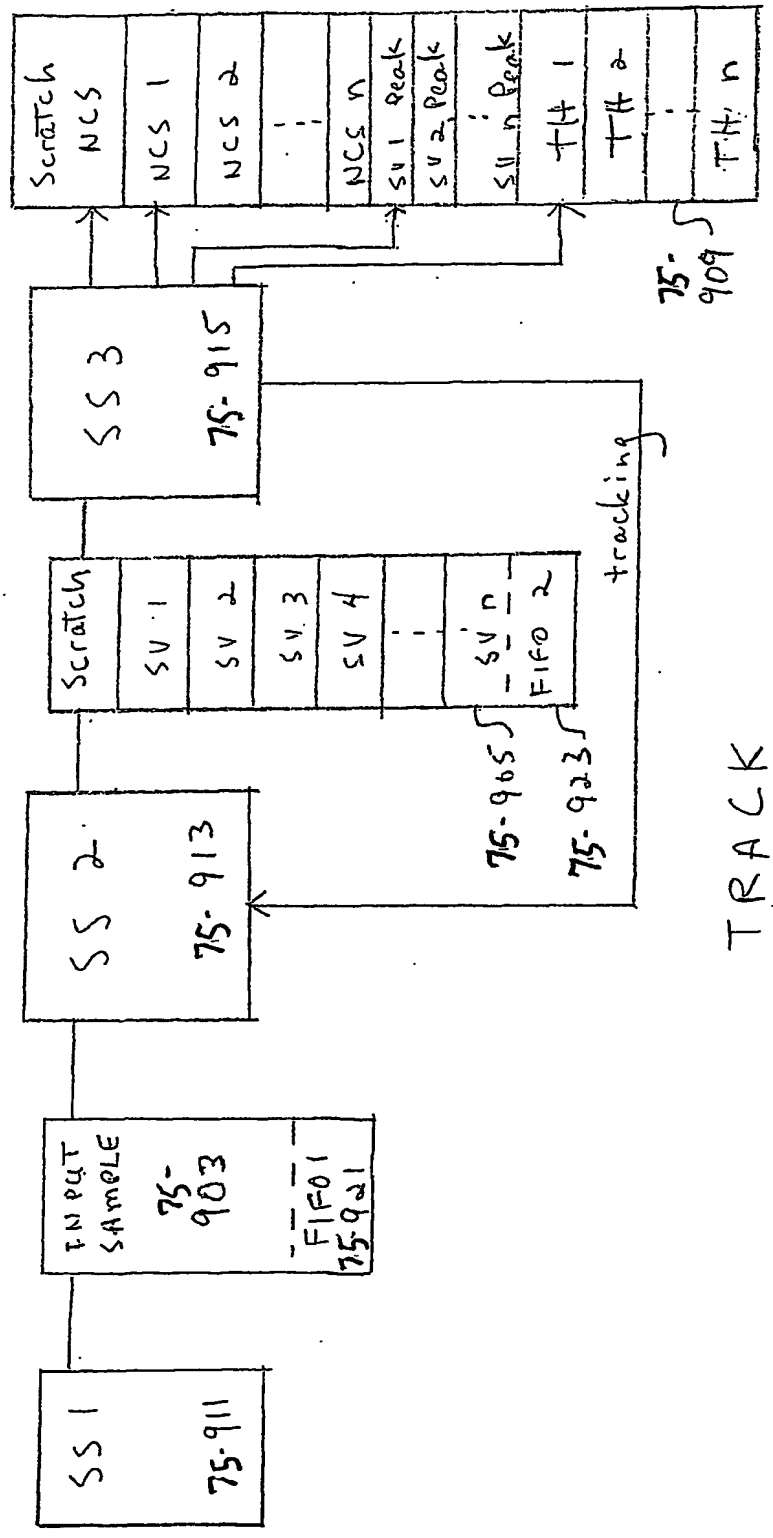

Scratch NCS
NCS 1
NCS 2
NCS n
SV 1 Peak
SV 2 Peak
SV n Peak
TH 1
TH 2
TH n
75-909
SS 3
75-915
tracking
Scratch
SV 1
SV 2
SV 3
SV 4
SV n
FIFO 2
75-905
75-923
SS 2
75-913
INPUT SAMPLE
75-903
FIFO1
75-921
SS 1
75-911
TRACK

FIG. 75

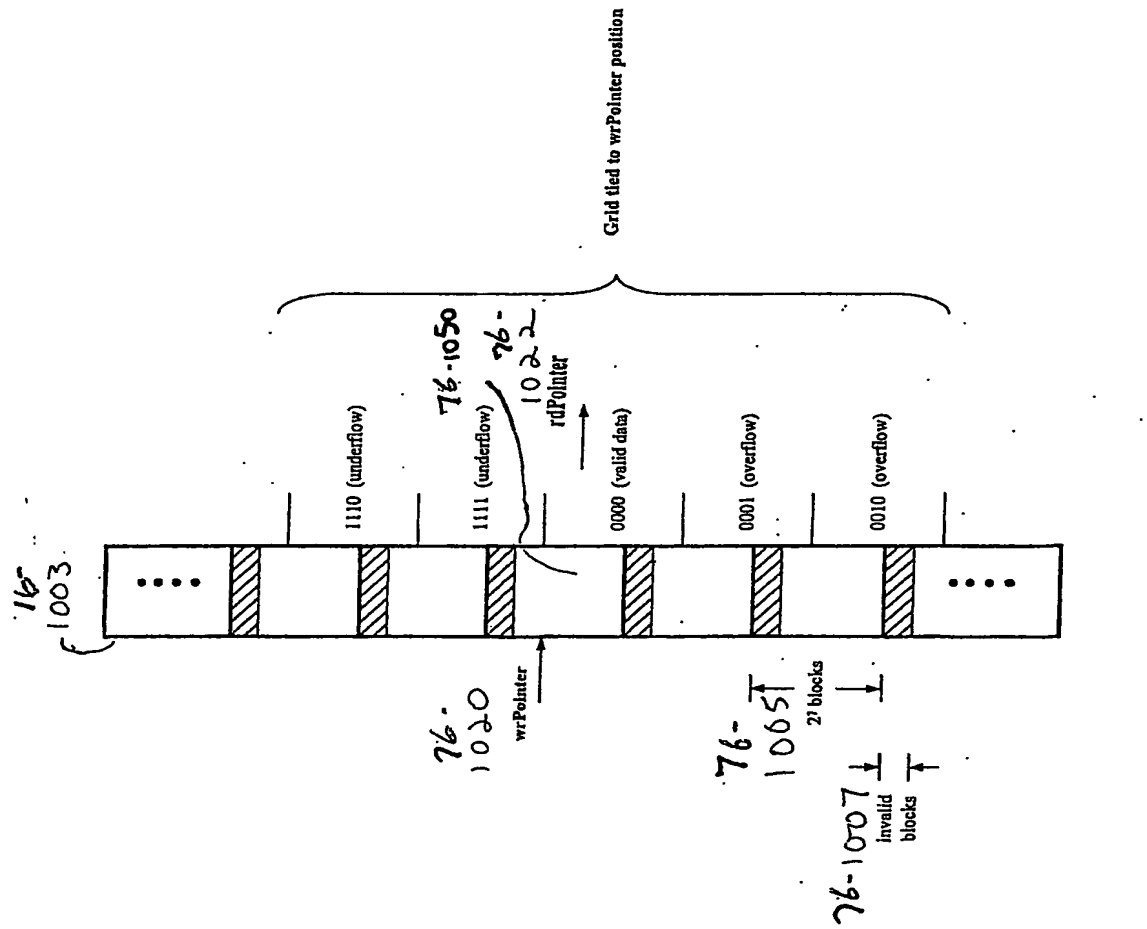

Grid tied to wrPointer position
1110 (underflow)
1111 (underflow)
0000 (valid data)
0001 (overflow)
0010 (overflow)
rdPointer
wrPointer
27 blocks
invalid blocks
16
1003
76-1050
76-1022
76-1020
76-1065
76-1007

FIG. 76

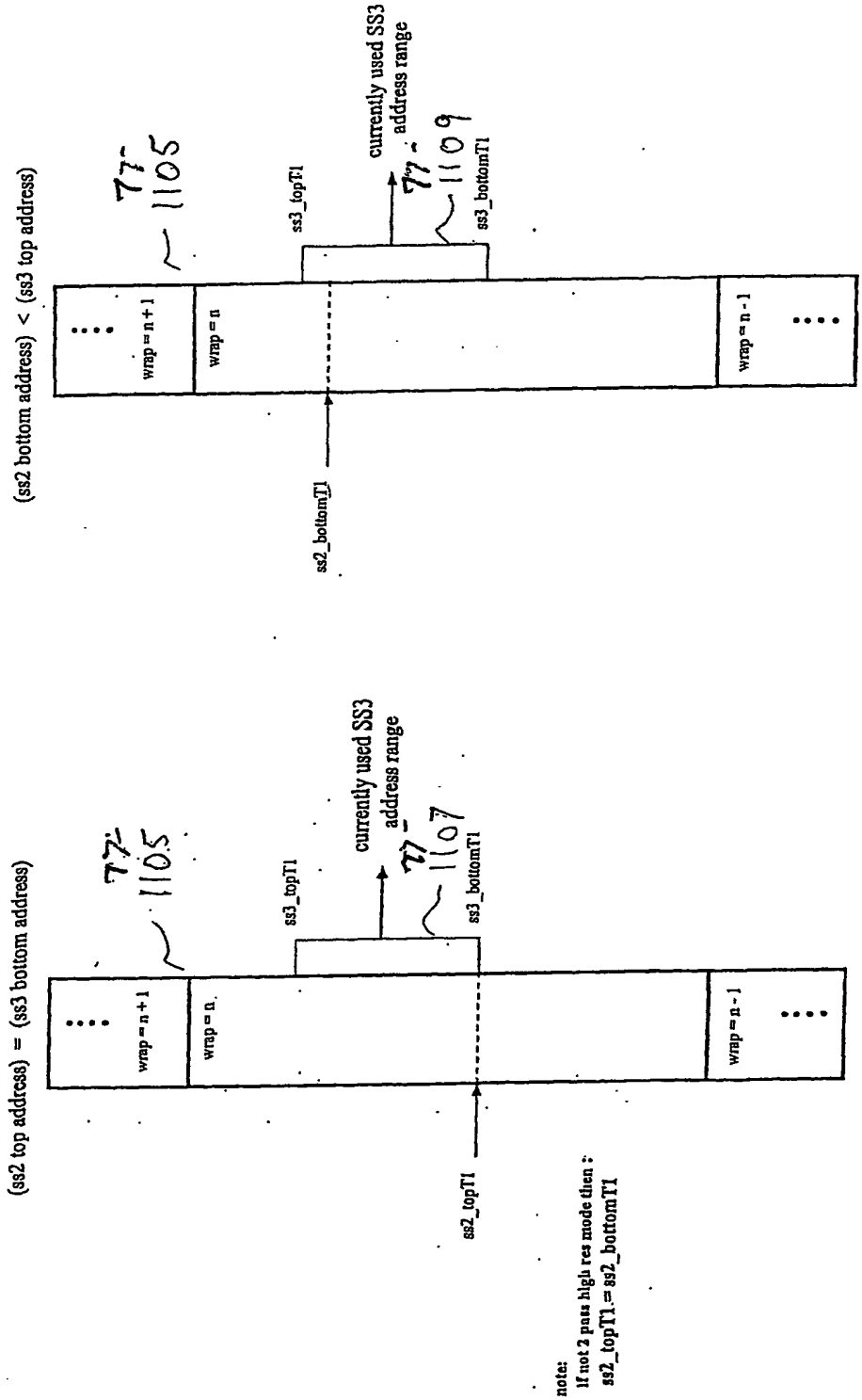

(ss2 bottom address) < (ss3 top address)
77-1105
wrap = n + 1
wrap = n
wrap = n - 1
ss3_topT1
ss3_bottomT1
currently used SS3 address range
77-1109
ss2_bottomT1
Underflow
(ss2 top address) = (ss3 bottom address)
77-1105
wrap = n + 1
wrap = n
wrap = n - 1
ss3_topT1
ss3_bottomT1
currently used SS3 address range
77-1107
ss2_topT1
Overflow
note:
If not 2 pass high res mode then :
ss2_topT1 = ss2_bottomT1

FIG. 77

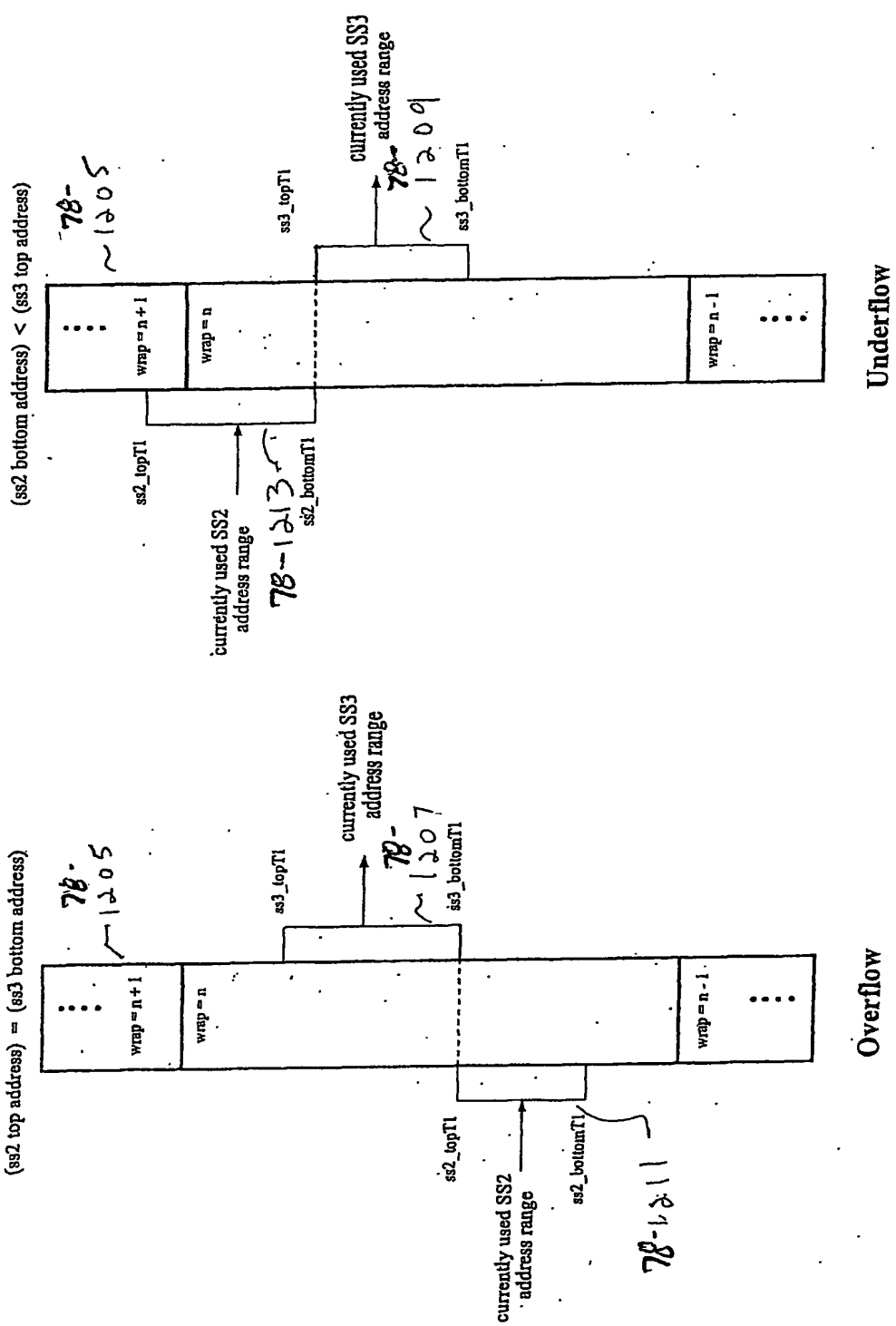

(ss2 bottom address) < (ss3 top address)
78-1205
wrap = n + 1
wrap = n
wrap = n - 1
ss3_topT1
ss3_bottomT1
currently used SS3 address range
78-1201
ss2_topT1
ss2_bottomT1
currently used SS2 address range
78-1213
Underflow
(ss2 top address) = (ss3 bottom address)
78-1205
wrap = n + 1
wrap = n
wrap = n - 1
ss3_topT1
ss3_bottomT1
currently used SS3 address range
78-1207
ss2_topT1
ss2_bottomT1
currently used SS2 address range
78-1211
Overflow

FIG. 78

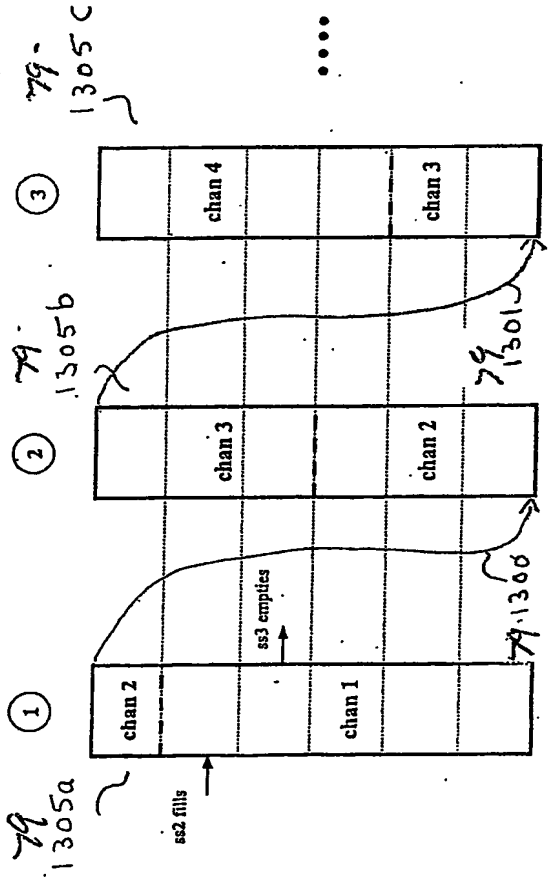

79-13055c
79-13055b
79-13305a
79-13301
79-13300
chan 4
chan 3
chan 3
chan 2
chan 2
chan 1
ss3 empties
ss2 fills
1
2
3

FIG. 79

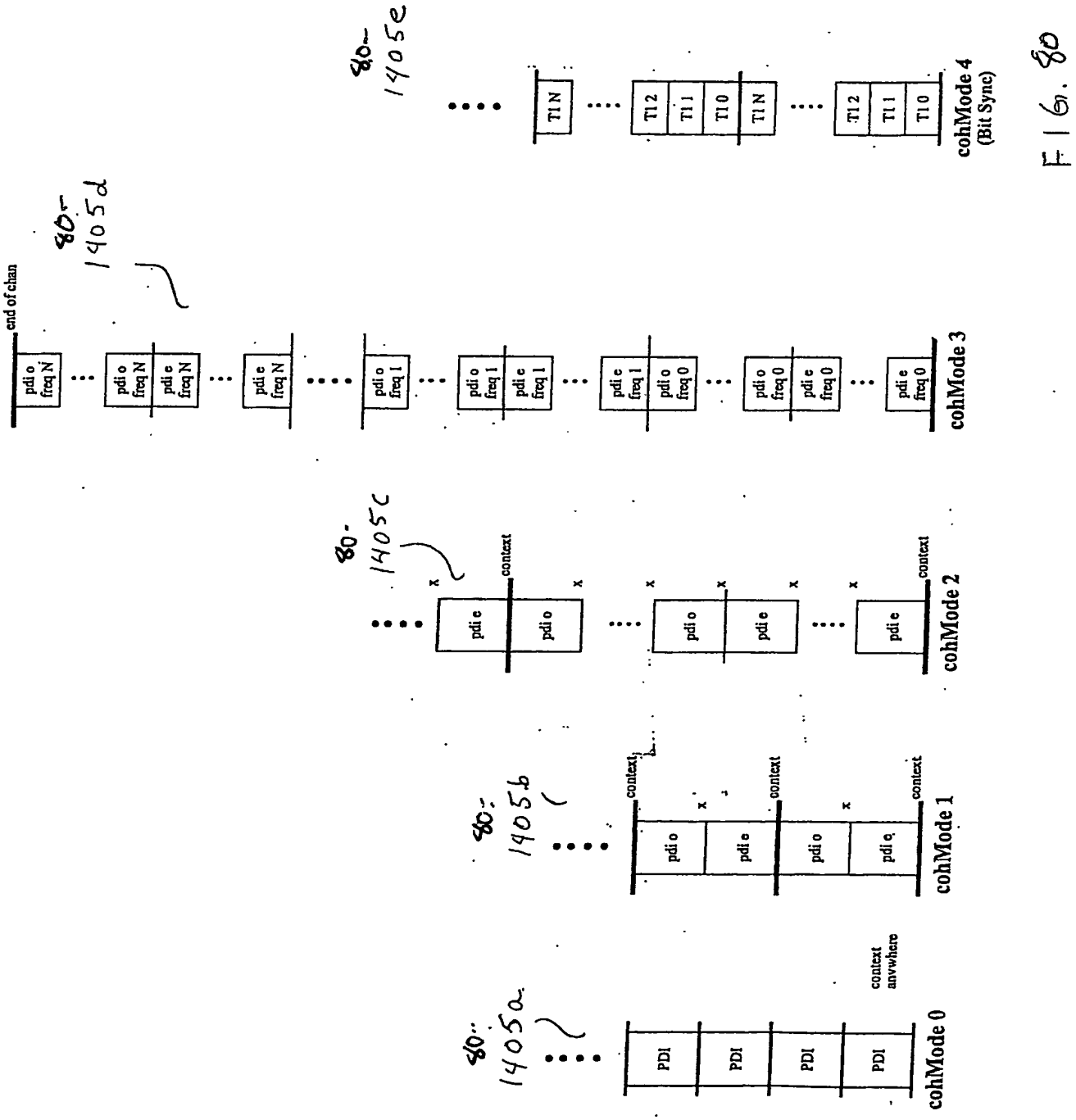
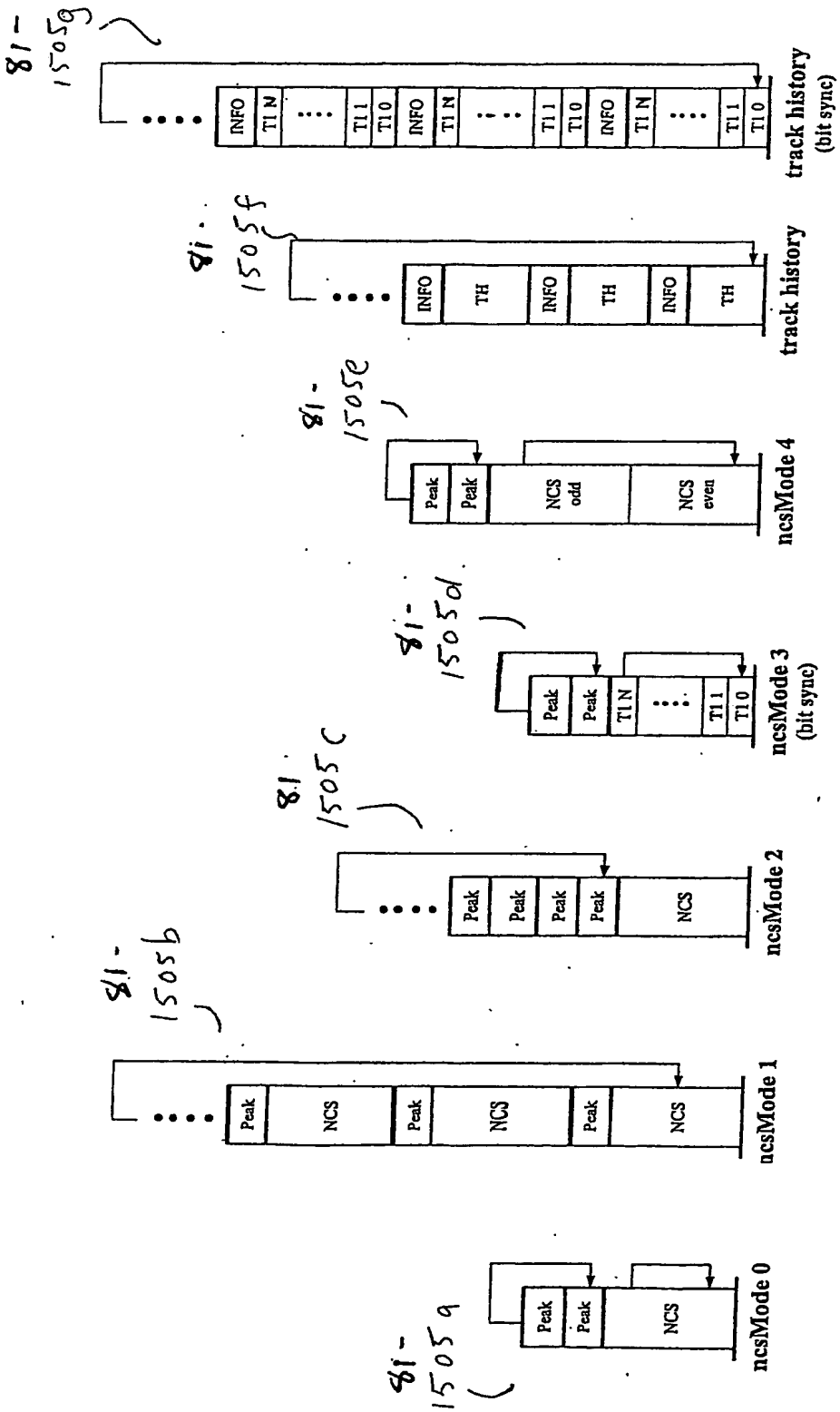

80-14050a
PDI
context anywhere
cohMode 0
80-14050b
context
pdi o
pdi e
cohMode 1
80-14050c
pdi e
pdi o
context
cohMode 2
80-14050d
end of chan
pdi o freq N
pdi e freq N
pdi o freq 1
pdi e freq 1
pdi o freq 0
pdi e freq 0
cohMode 3
80-14050e
T1 N
T1 2
T1 1
T1 0
cohMode 4
(Bit Sync)

FIG. 80


81-15059
INFO
T1 N
T1 1
T1 0
track history (bit sync)
81-15055
INFO
TH
track history
81-15052
Peak
NCS odd
NCS even
ncsMode 4
81-15051
T1 N
T1 1
T1 0
ncsMode 3 (bit sync)
81-15050
NCS
ncsMode 2
81-15056
ncsMode 1
81-15059
ncsMode 0

F16, 81

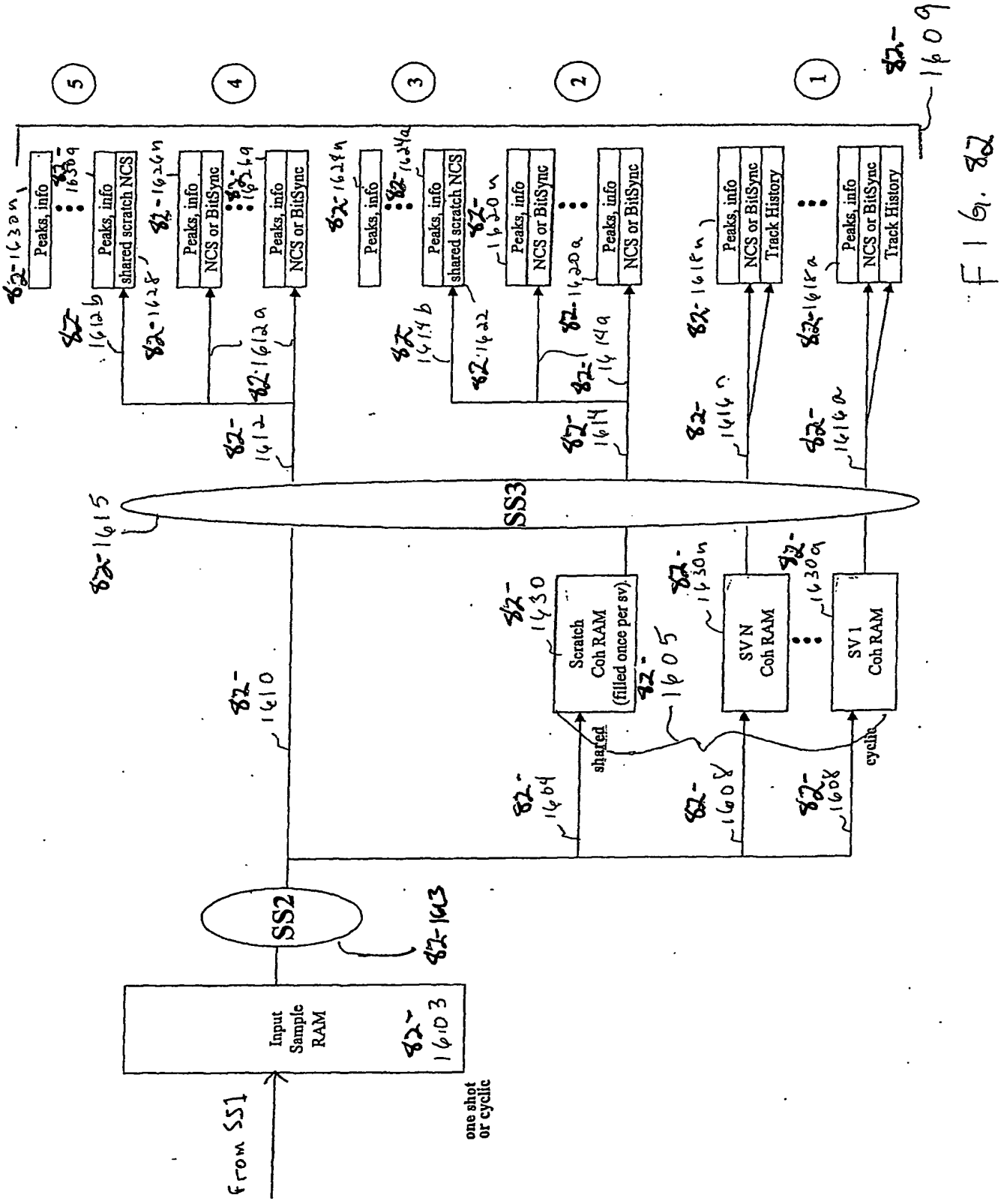

5
4
3
2
1
Peaks, info
shared scratch NCS
NCS or BitSync
Track History
SS3
SS2
Scratch Coh RAM (filled once per sv)
SV N Coh RAM
SV 1 Coh RAM
Input Sample RAM
shared
cyclic
from SS1
one shot or cyclic

FIG. 82

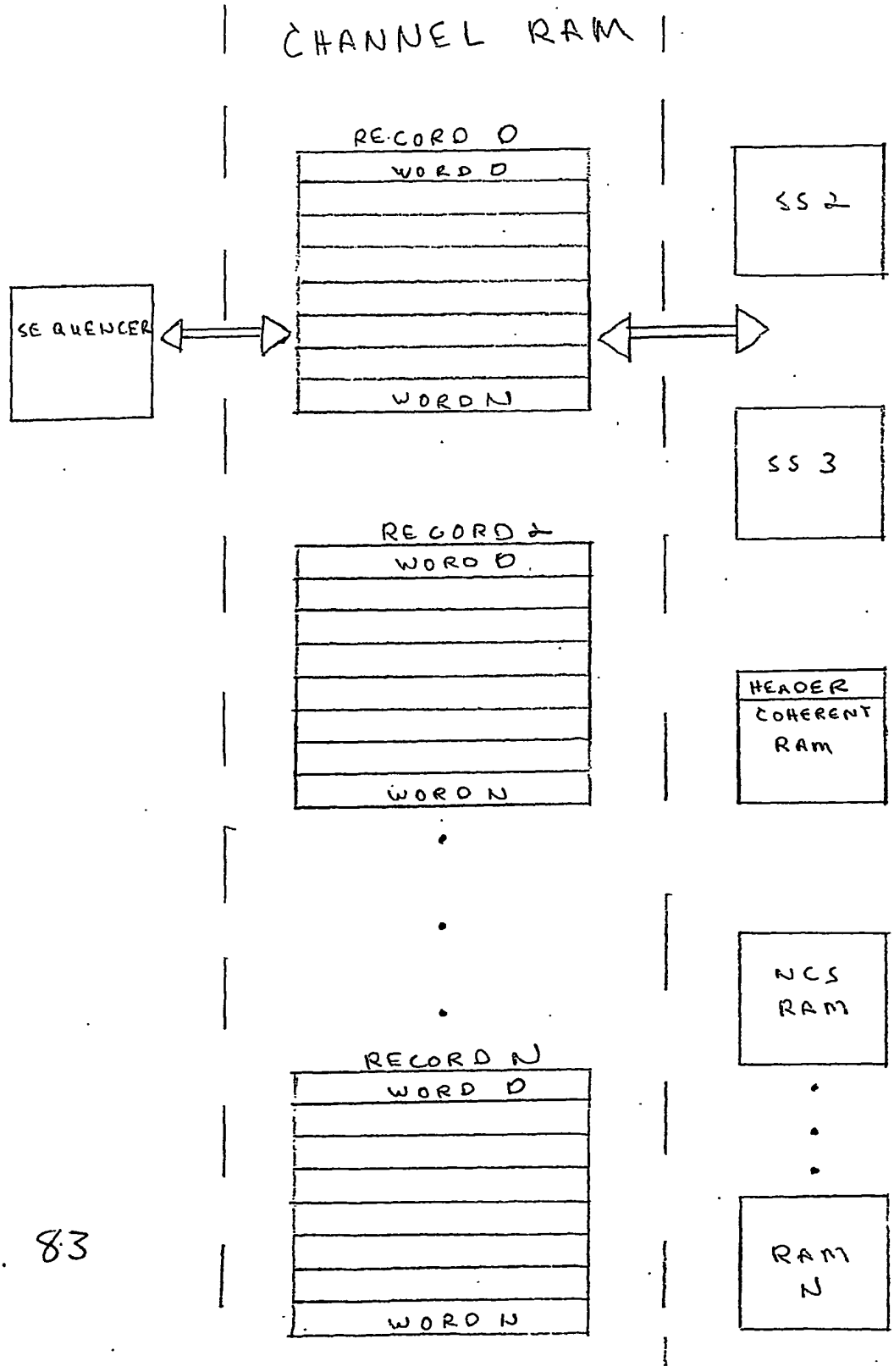

CHANNEL RAM
RECORD 0
WORD 0
WORD N
SEQUENCER
SS 2
SS 3
RECORD 2
WORD 0
WORD N
HEADER
COHERENT
RAM
NCS
RAM
RECORD N
WORD 0
WORD N
RAM
N

FIG. 83

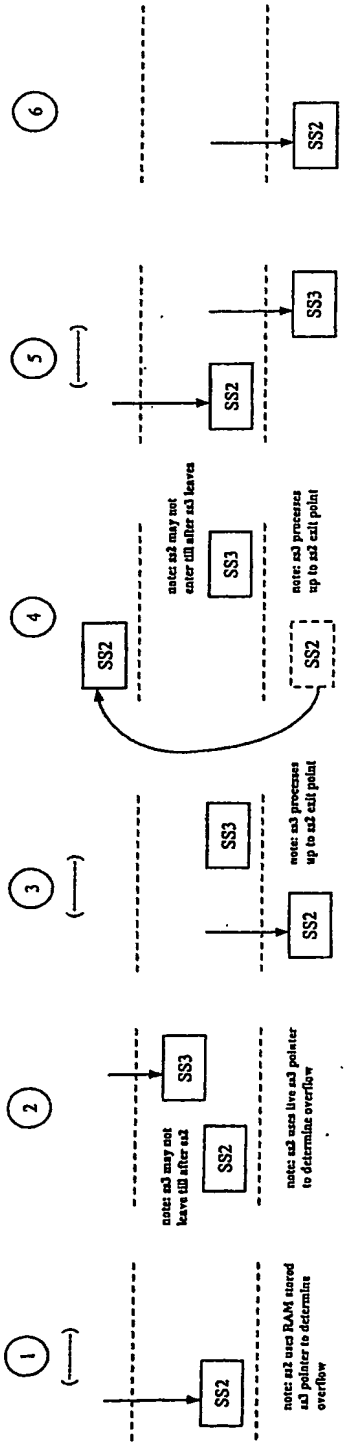

1
SS2
note: ss2 uses RAM stored ss3 pointer to determine overflow
2
SS3
SS2
note: ss3 may not leave till after ss2
note: ss3 uses live ss3 pointer to determine overflow
3
SS3
SS2
note: ss3 processes up to ss2 exit point
4
SS2
note: ss2 may not enter till after ss3 leaves
SS3
SS2
note: ss3 processes up to ss2 exit point
5
SS2
SS3
6
SS2

FIG. 84

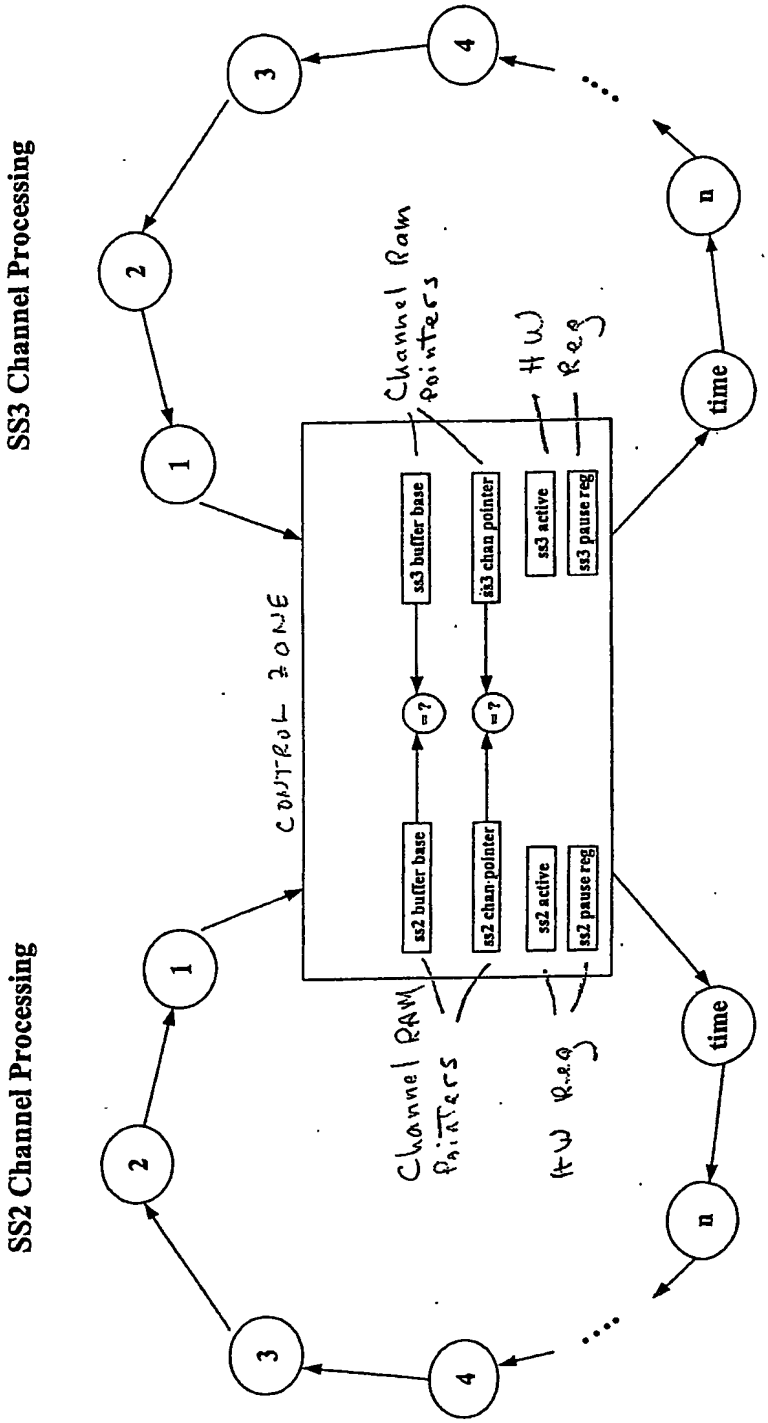

SS3 Channel Processing
SS2 Channel Processing
CONTROL ZONE
Channel RAM Pointers
HW Reg
ss3 buffer base
ss3 chan pointer
ss3 active
ss3 pause reg
ss2 buffer base
ss2 chan pointer
ss2 active
ss2 pause reg
=?
=?
Channel RAM Pointers
HW Reg
time
time
1
2
3
4
n

FIG. 85

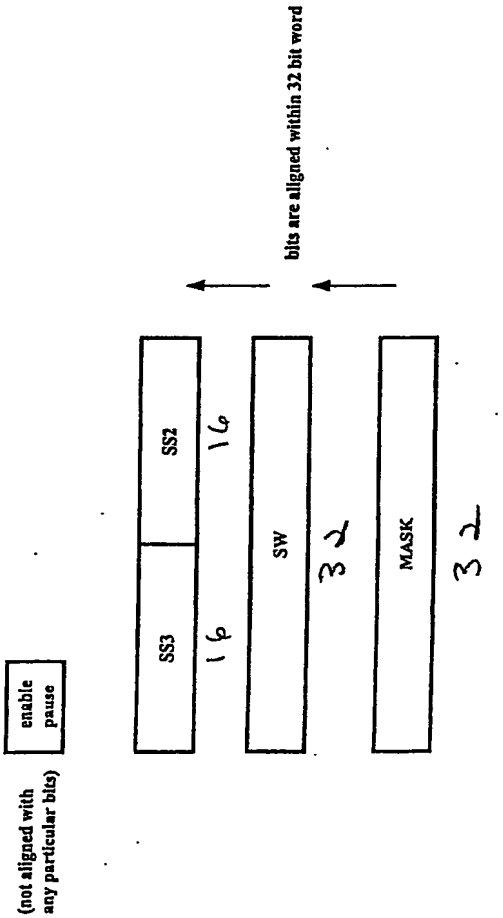

bits are aligned within 32 bit word
SS2
SS3
16
16
SW
32
MASK
32
enable pause
(not aligned with any particular bits)

FIG 86

Semaphores 1st half:
ncsShutOffSW
ss3PhaseAdjustSW
hwIdSW
aldingNCOSW
normASErrorSW
xcorrASErrorSW
xcorrParamLateSW
ncsOvrflwSW
fifoOvrflwSW
ss3100mutSW
ss3PDDoneSW
ss3ChDoneSW
ncsCompleteSW
ss3OnSW
Semaphores 2nd half:
ncsShutOffHW
ss3PhaseAdjustHW
hwIdHW
aldingNCOHW
normASErrorHW
xcorrASErrorHW
xcorrParamLateHW
ncsOvrflwHW
fifoOvrflwHW
ss3100mutHW
ss3PDDoneHW
ss3ChDoneHW
ncsCompleteHW
ss3Active
ss3OnSS3
ss2OnSS3
Interupt Enables: (MASK)
enNCSShutOffHW
enSS3PhaseAdjust
enIntrptHWTL
enIntrptAlding
enIntrptNormASError
enIntrptXcorrASError
enIntrptXcorrParamLate
enIntrptNCSOvrflw
enIntrptFFTovrflw
enIntrpt100mut
enIntrptSS3PDI
enIntrptSS3ChanDn
enIntrptNCSComplete
enIntrptSS3Pause
enIntrptSS3On
enIntrptSS2X
enIntrptAdjustEndMs
enIntrptSS2Xcorr
enIntrptCohSat
enIntrptFIFO1
enIntrptSS2ChanDn
enIntrptSS2Pause
enIntrptSS2On
ss2OnSW
ss2ChDoneSW
fifo1OvrflwSW
cohSatSW
ss2XcorrSW
ss2AdjEndMsSW
ss2PhaseAdjustSW
ss2ParamErrorSW
ss2ParamErrorHW
ss2PhaseAdjustHW
ss2AdjEndMsHW
ss2XcorrHW
cohSatHW
fifo1OvrflwHW
ss2ChDoneHW
ss2Active
ss3OnSS2
ss2OnSS2

FIG. 87

HW Controlled	SW Controlled	Enable for stalls
HW SS2 on	SW SS2 On	
HW SS3 on	SW SS3 On	
HW SS2 Active		
HW SS3 Active		
HW set SS2 finished chan	SW clear SS2 finished chan	
HW set SS3 finished chan	SW clear SS3 finished chan	ss2 wait
HW set SS3 pdi done	SW clear SS3 pdi done	ss2 wait, ss3 wait
HW FIFO1 overflow	SW clear FIFO1 overflow	
HW SS2 coh accum clip	SW clear SS2 coh accum clip	

FIG. 68

- SS2: FIFO1 chan initiation overflow detect
  - shut down SS2 (ss2OnSS2 = 0) and SS3 (ss3OnSS2 = 0)
  - set fifo1Overflw semaphore and interrupt (if enabled)

- SS3: FIFO2 errors - none identified that would cause termination

- SS3: NCS timeout – occurs when ncsCount reaches greater than ncsCountMod
  - if ncsStopNormalEn ==1 then shut down SS2 (ss2OnSS3 = 0) and SS3 (ss3OnSS3 = 0)
  - set ncsComplete semaphore and interrupt (if enabled)
  - bit sync mode : allow all T1 offsets to complete before terminating channel or setting semaphore
  - odd/even/multFreq: allow all runs to complete before terminating channel or setting semaphore

- SS3: NCS overflow prevention or autoscale termination event, very strong signal detect
  - ncs overflow occurs if all autoscale bits are used up and another scale would be required on next pass
  - autoscale termination event occurs if ncs autoscale adjust detected on current PDI and (ncsCount < ncsASStop[currentAS])
  - stop NCS accumulations
    - if ncsStopEarlyEn == 1 then:
      1) shut down SS2 (ss2OnSS3 = 0) and SS3 (ss3OnSS3 = 0)
      2) set ncsComplete semaphore and interrupt (if enabled)
    - otherwise disable ncs accumulation and peak generation for ncs duration and then start up again
  - set ncsComplete semaphore and interrupt (if enabled)
  - bit sync mode : allow all T1 offsets to complete before terminating channel or setting semaphore
  - odd/even/multFreq: allow all runs to complete before terminating channel or setting semaphore

- SS3: SW command asynchronous ncsShutOff
  - allows SW clean way to turn off channel
  - commanded by semaphore
  - SS3 complete current context and upon channel shut down it resets ss2OnSS3 and ss3OnSS3
    - this allows cleaning out from shared FIFO2 of any data generated by the channel

FIG. 89

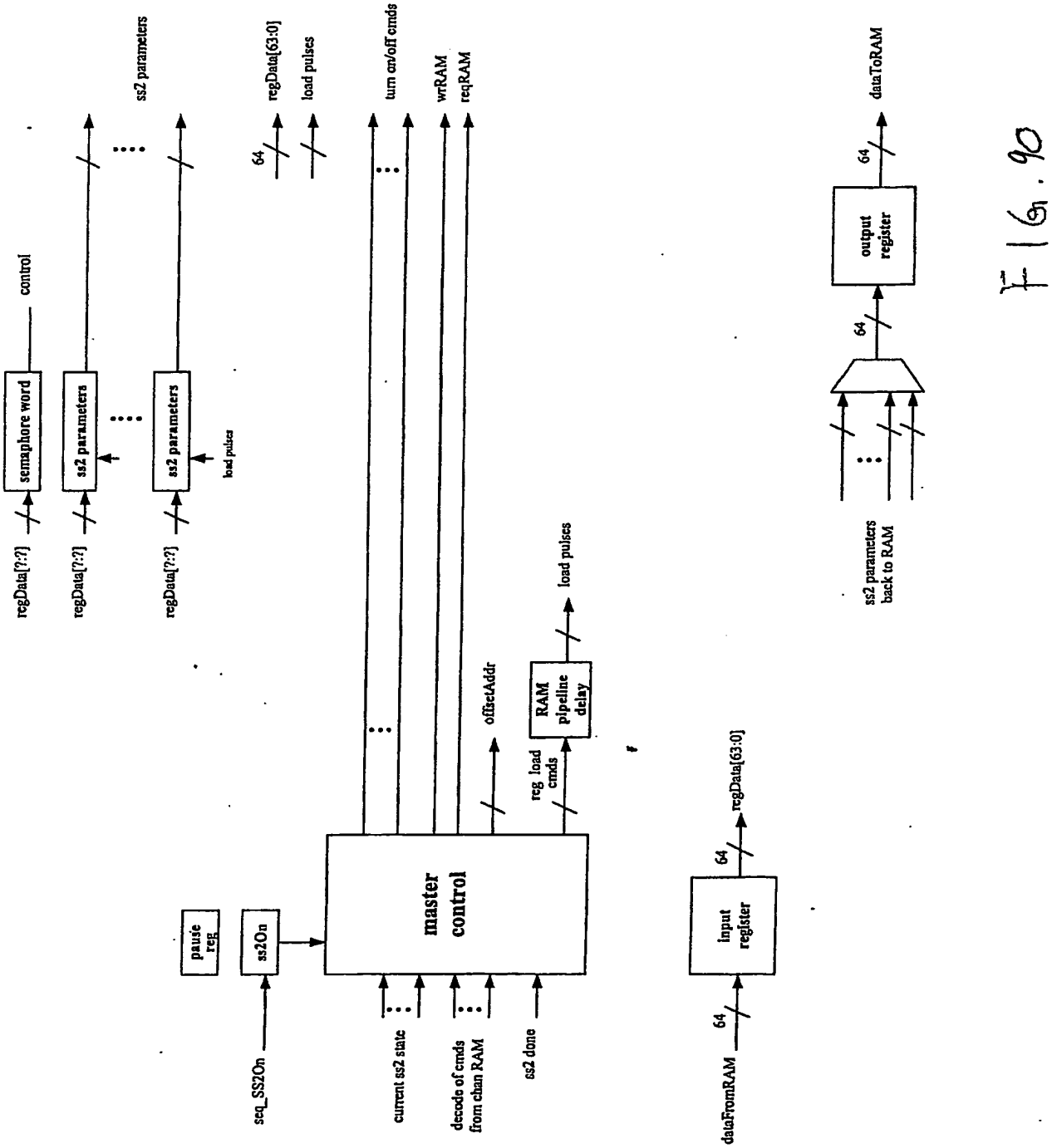

control
semaphore word
ss2 parameters
ss2 parameters
load pulses
ss2 parameters
regData[?:?]
regData[?:?]
regData[?:?]
64
regData[63:0]
load pulses
turn on/off cmds
wrRAM
reqRAM
pause reg
ss2On
seq_SS2On
master control
current ss2 state
decode of cmds from chan RAM
ss2 done
offsetAddr
reg load cmds
RAM pipeline delay
load pulses
64
input register
regData[63:0]
dataFromRAM
64
output register
dataToRAM
64
ss2 parameters back to RAM

FIG. 90

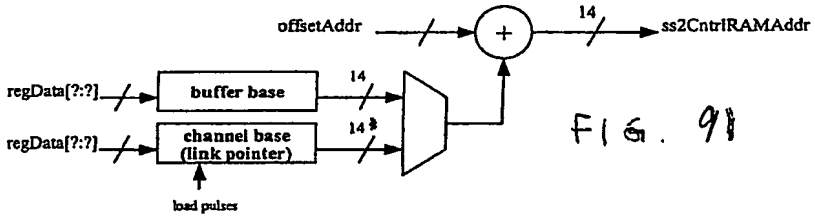

offsetAddr
14
ss2CntrlRAMAddr
regData[?:?]
buffer base
14
regData[?:?]
channel base
(link pointer)
14
load pulses

FIG. 9I

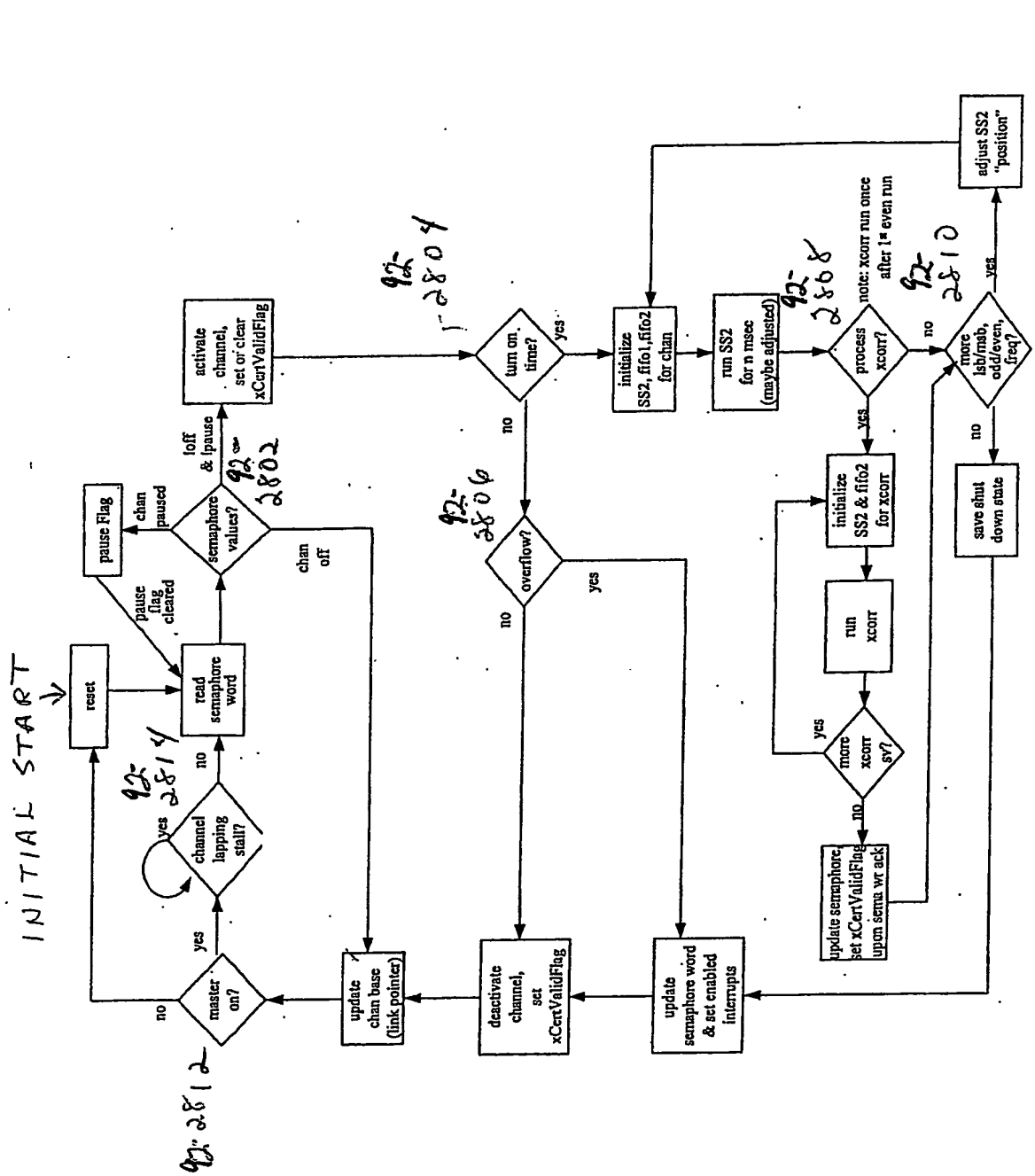

INITIAL START
reset
read semaphore word
92-2814
channel lapping stall?
yes
no
92-2812
master on?
yes
no
update chan base (link pointer)
pause Flag
pause flag cleared
chan paused
semaphore values?
chan off
loff & lpause
92-2802
activate channel, set or clear xCertValidFlag
92-2804
turn on time?
yes
no
deactivate channel, set xCertValidFlag
92-2806
overflow?
yes
no
update semaphore word & set enabled interrupts
initialize SS2, fifo1,fifo2 for chan
run SS2 for n msec (maybe adjusted)
92-2808
process xcorr?
yes
no
note: xcorr run once after 1st even run
initialize SS2 & fifo2 for xcorr
run xcorr
more xcorr sv?
yes
no
update semaphore, set xCertValidFlag upon sema wr ack
92-2810
more lsb/msb, odd/even, freq?
yes
no
adjust SS2 "position"
save shut down state

FIG. 92

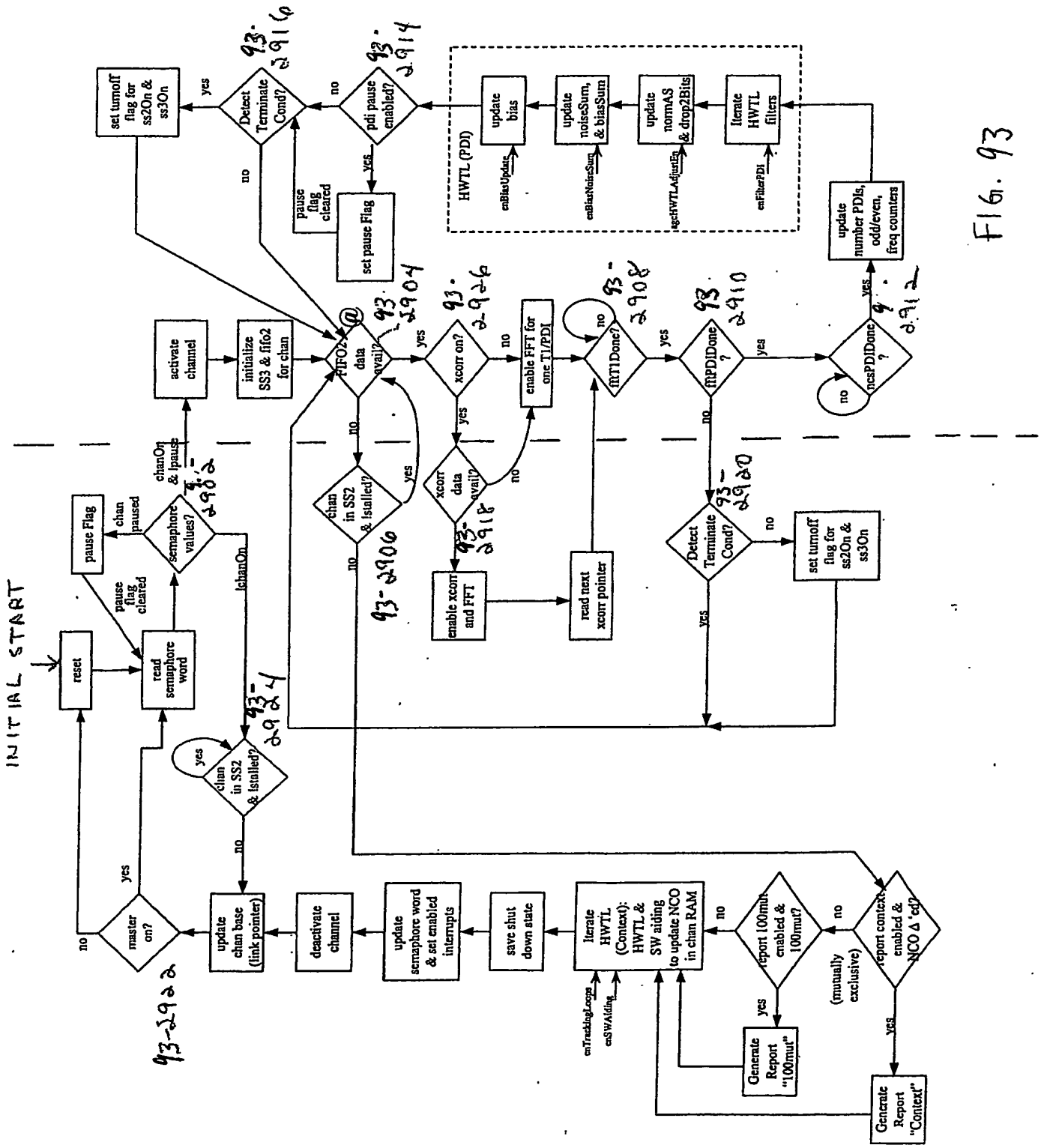

INITIAL START
reset
read semaphore word
pause Flag
chan paused
pause flag cleared
semaphore values?
chanOn & !pause
activate channel
initialize SS3 & fifo2 for chan
chanOn
chan in SS2 & istalled?
master on?
update chan base (link pointer)
deactivate channel
update semaphore word & set enabled interrupts
save shut down state
Iterate HWTL (Context): HWTL & SW aiding to update NCO in chan RAM
enTrackingLoops
enSWAiding
report 100mut enabled & 100mut?
Generate Report "100mut"
report context enabled & NCO Δ 'ed?
Generate Report "Context"
(mutually exclusive)
fifo2 data avail?
xcorr on?
enable FFT for one T1/PDI
fft1Done?
fftPDIDone?
ncsPDIDone?
update number PDIs, odd/even, freq counters
xcorr data avail?
enable xcorr and FFT
read next xcorr pointer
chan in SS2 & istalled?
Detect Terminate Cond?
set turnoff flag for ss2On & ss3On
pdi pause enabled?
pause flag cleared
set pause Flag
HWTL (PDI)
update bias
update noiseSum, & biasSum
update normAS & drop2Bits
Iterate HWTL filters
enBiasUpdate
enBiasNoiseSum
enHWTLAdjustBit
enFilterPDI
93-2902
93-2904
93-2906
93-2908
93-2910
93-2912
93-2914
93-2916
93-2918
93-2920
93-2922
93-2924
93-2926

FIG. 93

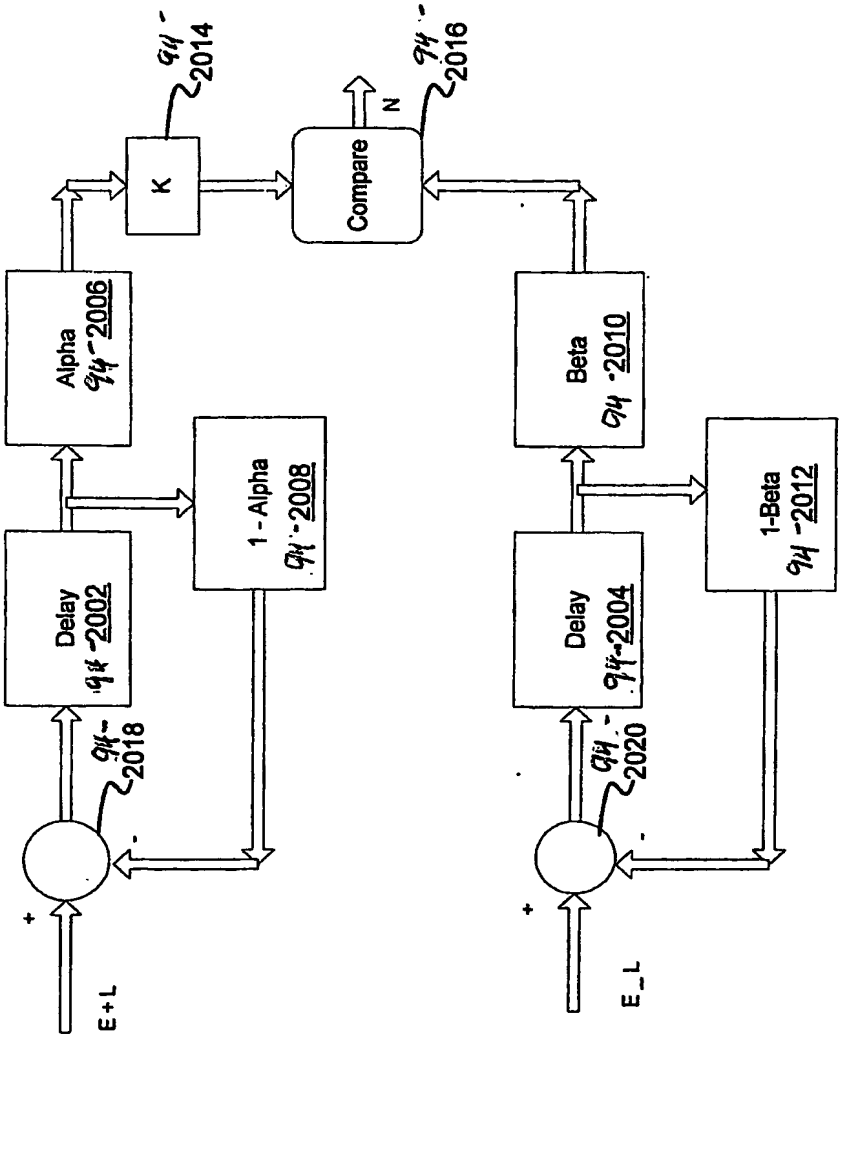

94 - 2014
94 - 2016
K
Compare
N
Alpha 94 - 2006
Beta 94 - 2010
1 - Alpha 94 - 2008
1-Beta 94 - 2012
Delay 94 - 2002
Delay 94 - 2004
94 - 2018
94 - 2020
+
-
+
-
E+L
E_L
2000

Figure 94

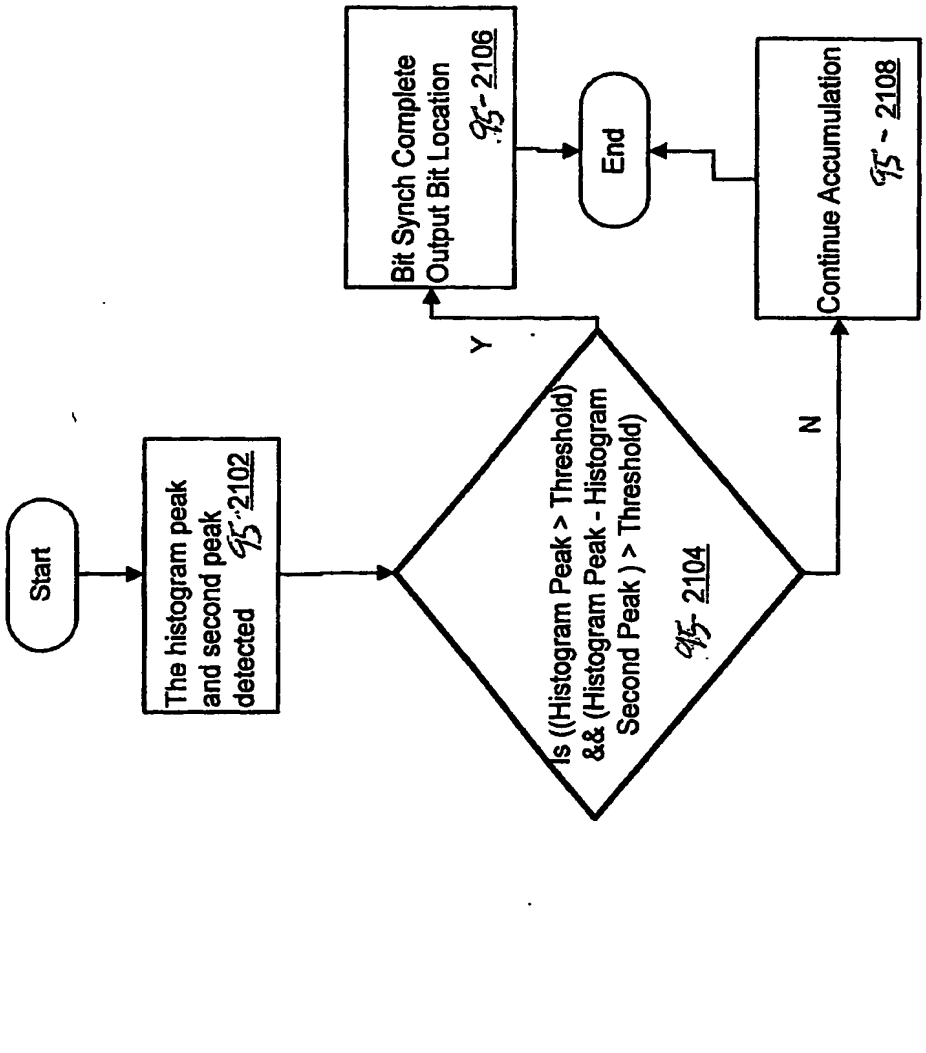

Start
The histogram peak and second peak detected 95-2102
Is ((Histogram Peak > Threshold) && (Histogram Peak - Histogram Second Peak) > Threshold) 95-2104
Y
N
Bit Synch Complete Output Bit Location 95-2106
End
Continue Accumulation 95-2108
95-2100

Figure 95

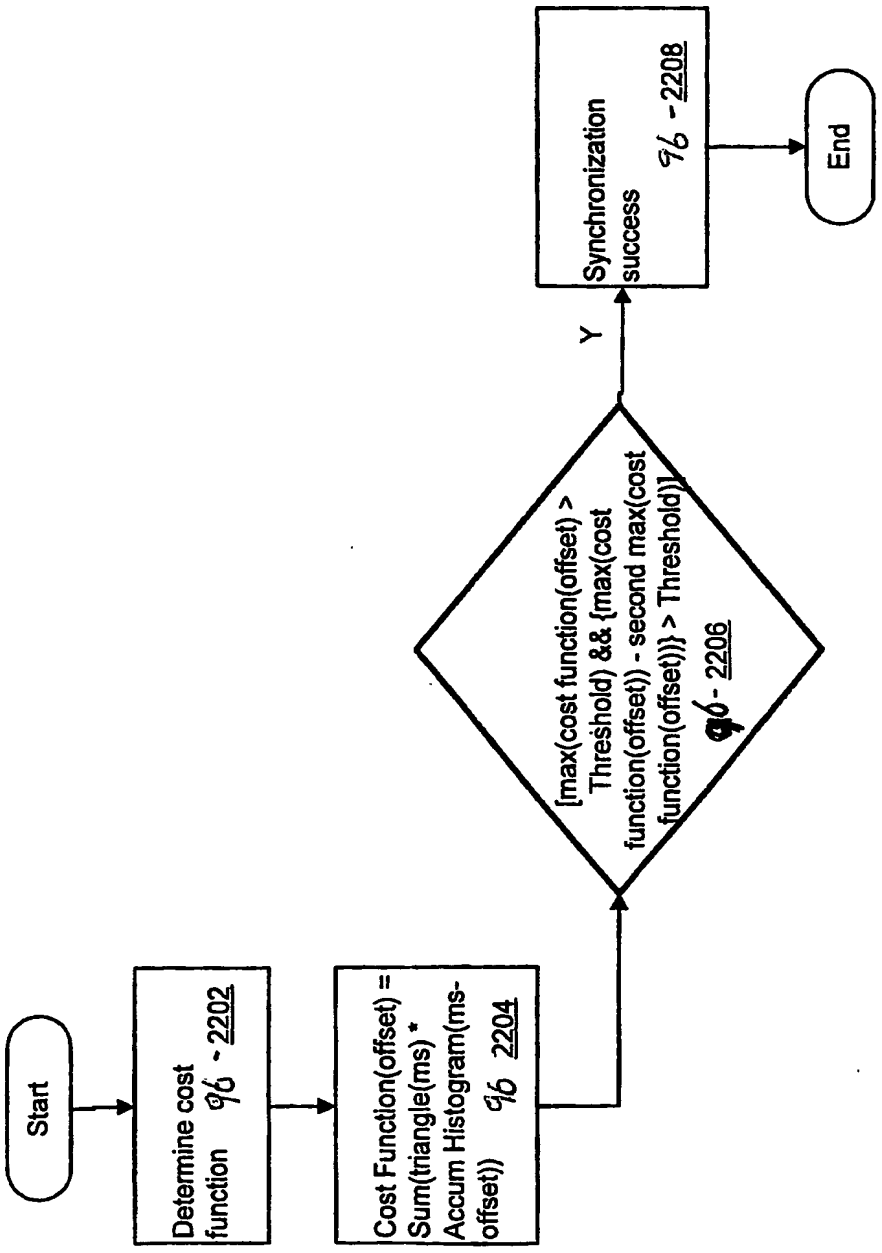

Start
Determine cost function 96 - 2202
Cost Function(offset) = Sum(triangle(ms) * Accum Histogram(ms-offset)) 96 2204
[max(cost function(offset) > Threshold) && {max(cost function(offset)) - second max(cost function(offset))} > Threshold)] 96 - 2206
Y
Synchronization success 96 - 2208
End
96 - 2200

Figure 96

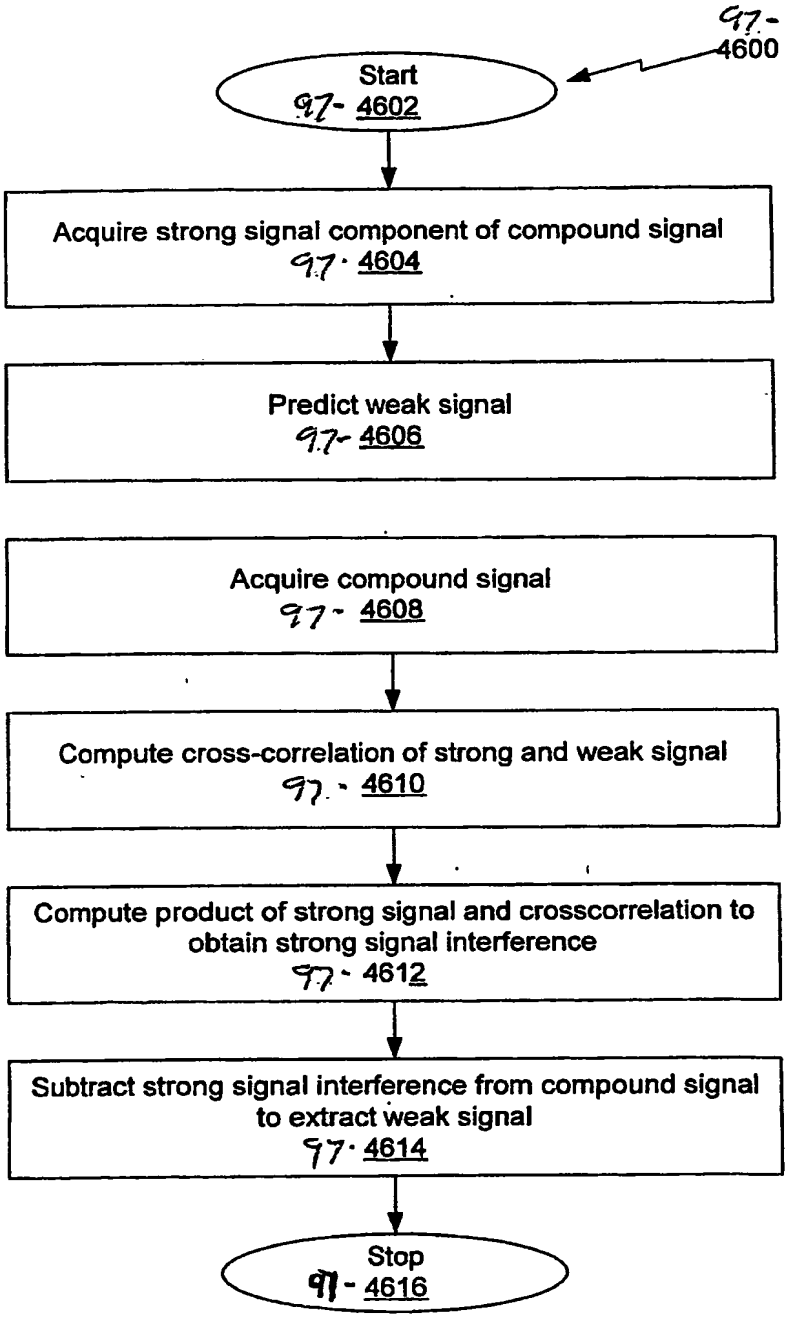

97-4600
Start
97- 4602
Acquire strong signal component of compound signal
97- 4604
Predict weak signal
97- 4606
Acquire compound signal
97- 4608
Compute cross-correlation of strong and weak signal
97- 4610
Compute product of strong signal and crosscorrelation to obtain strong signal interference
97- 4612
Subtract strong signal interference from compound signal to extract weak signal
97- 4614
Stop
97- 4616

FIG. 97

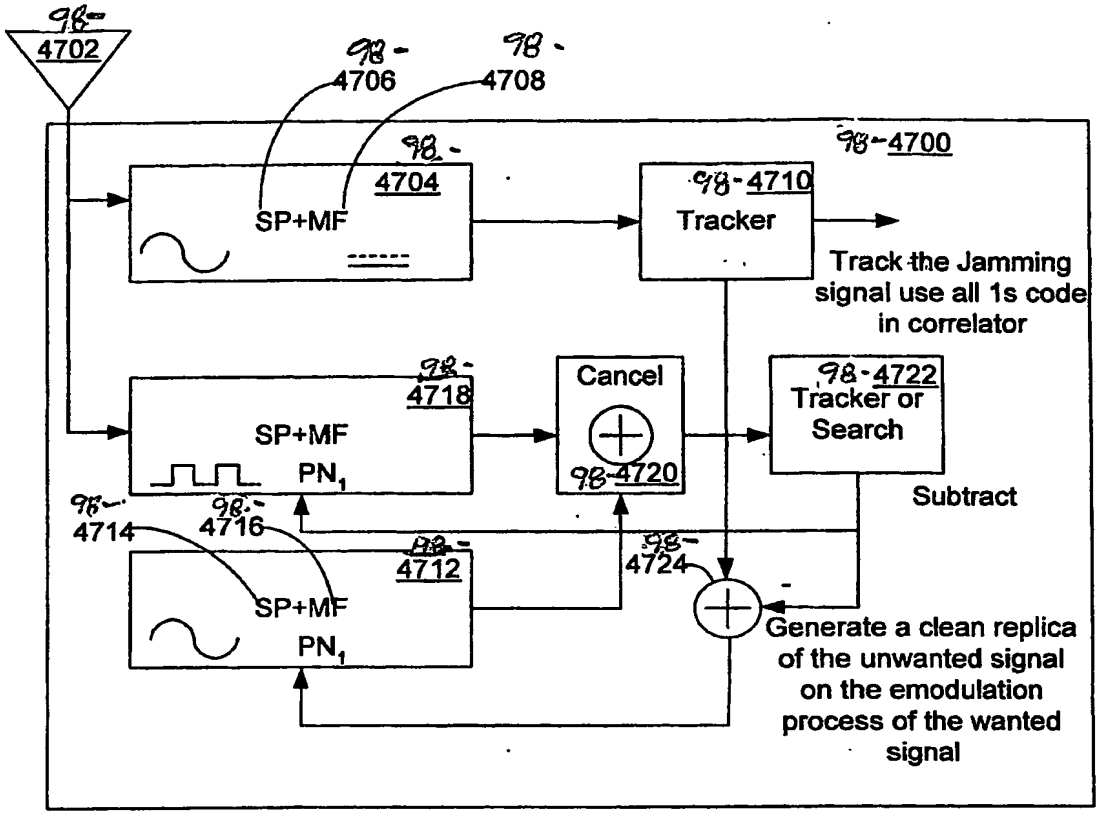

98-4702
98-4706
98-4708
98-4700
98-4704
SP+MF
98-4710
Tracker
Track the Jamming signal use all 1s code in correlator
98-4718
SP+MF
PN1
Cancel
98-4720
98-4722
Tracker or Search
Subtract
98-4714
98-4716
98-4712
SP+MF
PN1
98-4724
Generate a clean replica of the unwanted signal on the emodulation process of the wanted signal

FIG. 98

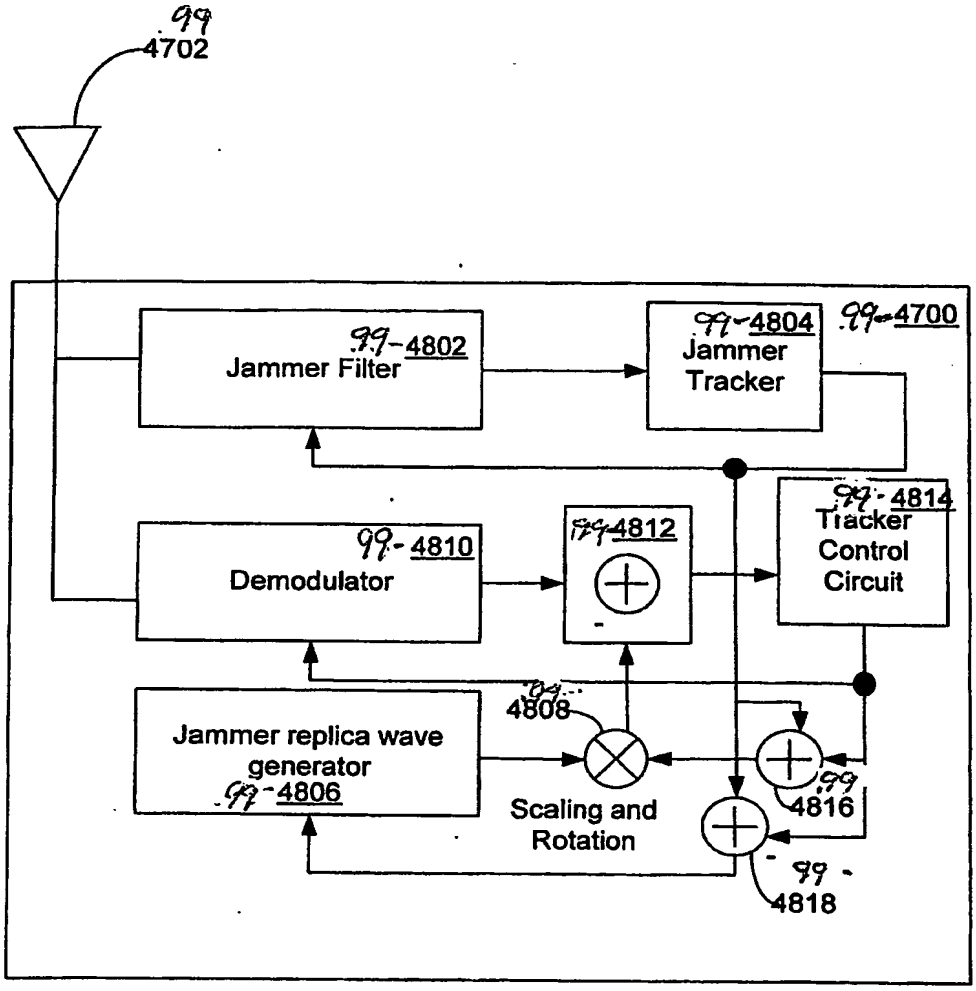

99
4702
99-4802
Jammer Filter
99-4804
Jammer
Tracker
99-4700
99-4814
Tracker
Control
Circuit
99-4810
Demodulator
99-4812
99
4808
Jammer replica wave
generator
99-4806
Scaling and
Rotation
99
4816
99
4818

FIG.
99.

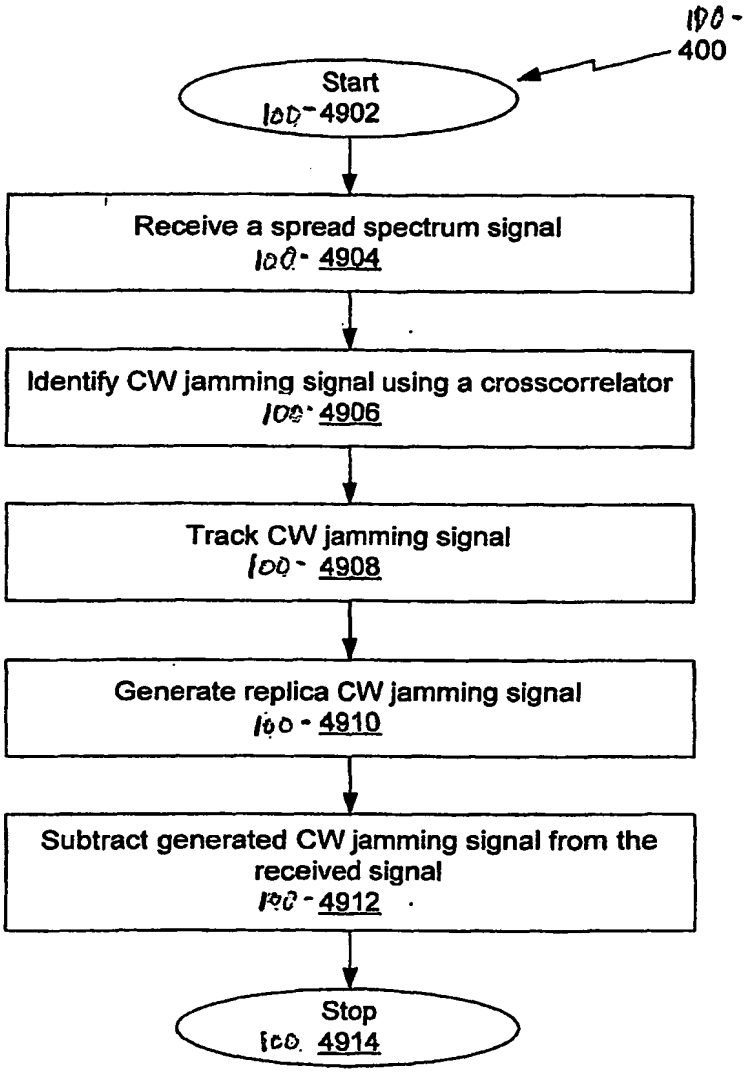

100 - 400
Start
100 - 4902
Receive a spread spectrum signal
100 - 4904
Identify CW jamming signal using a crosscorrelator
100 - 4906
Track CW jamming signal
100 - 4908
Generate replica CW jamming signal
100 - 4910
Subtract generated CW jamming signal from the received signal
100 - 4912
Stop
100 4914

FIG. 100

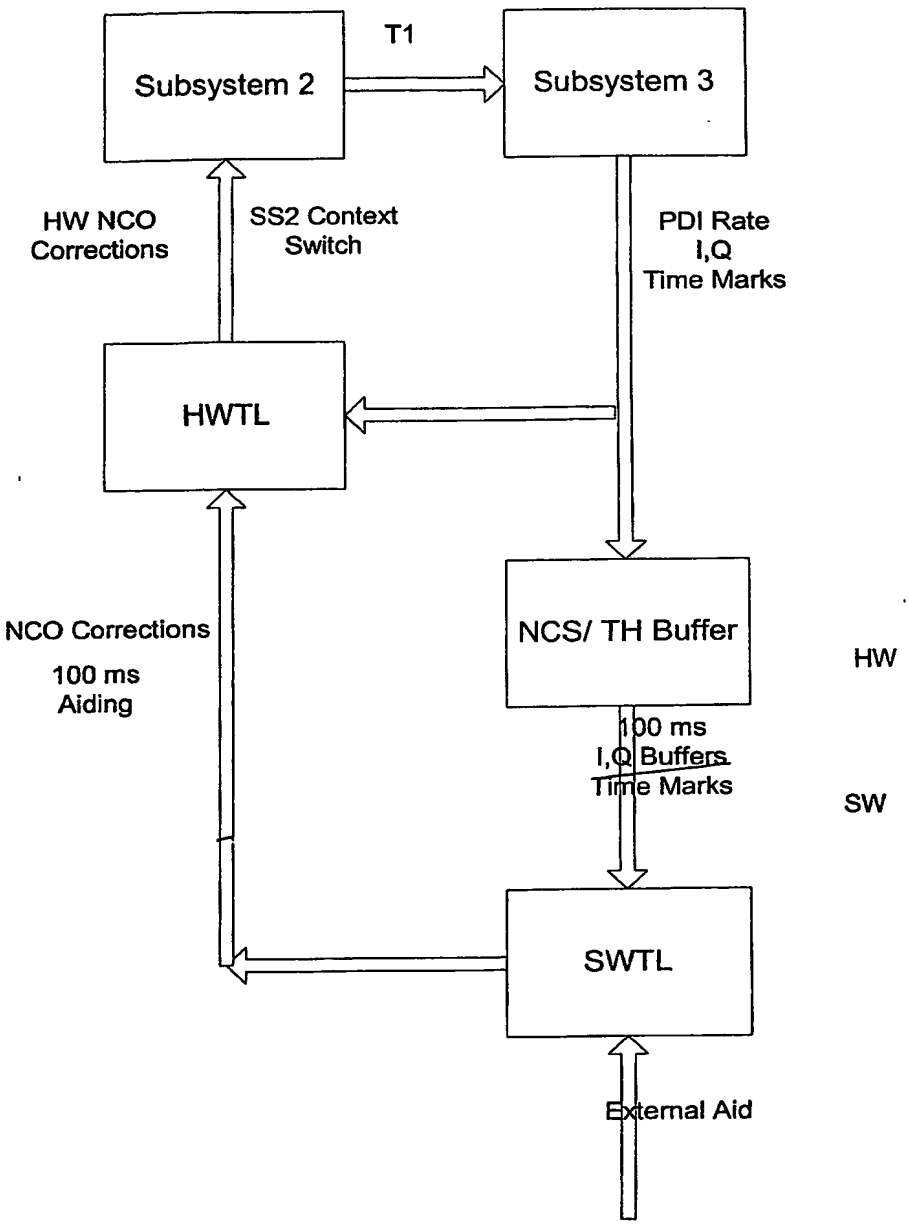

Subsystem 2
T1
Subsystem 3
HW NCO Corrections
SS2 Context Switch
PDI Rate I,Q Time Marks
HWTL
NCO Corrections
100 ms Aiding
NCS/ TH Buffer
HW
100 ms I,Q Buffers Time Marks
SW
SWTL
External Aid

Figure 101

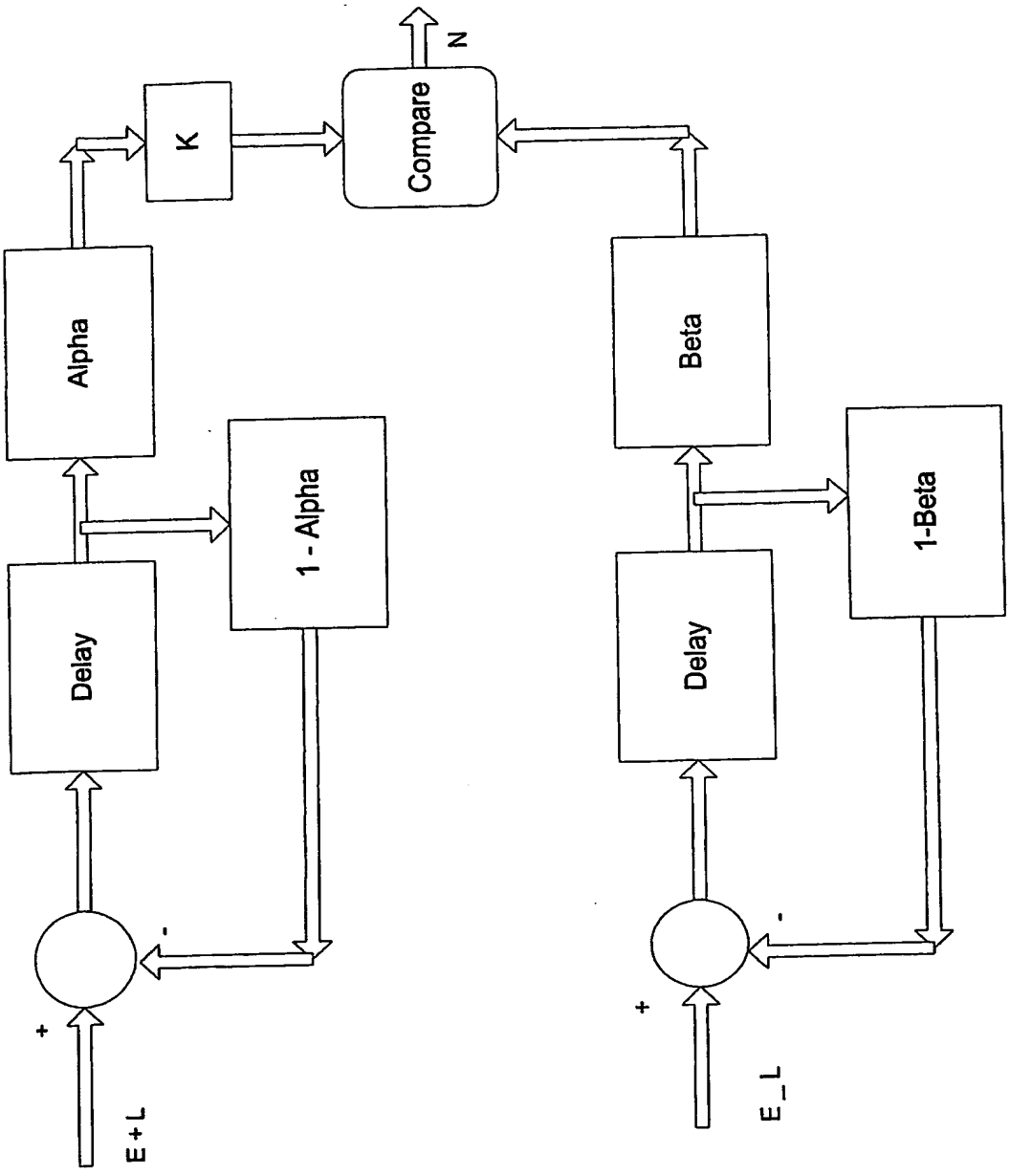


Figure 102

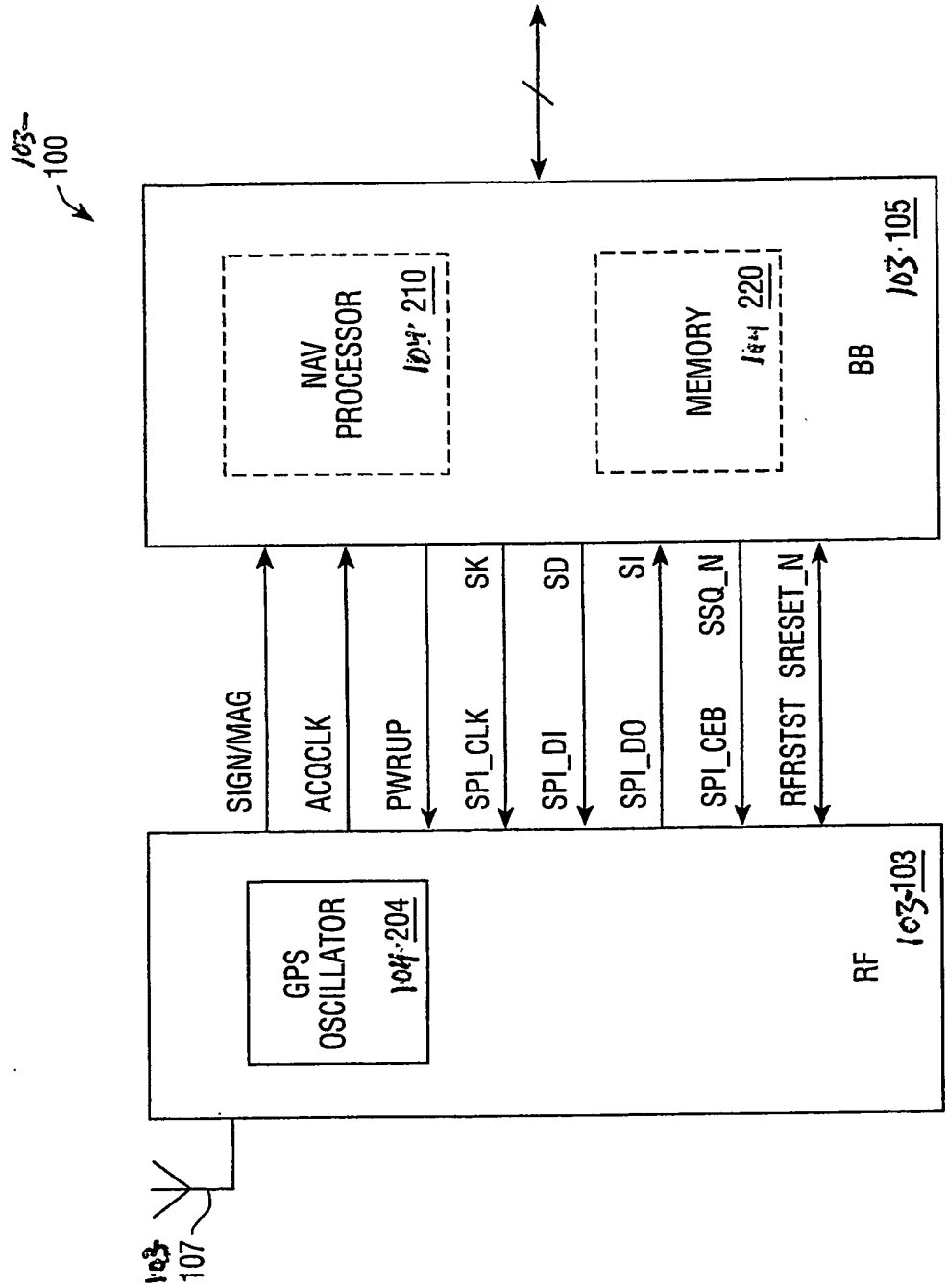

103-100
NAV PROCESSOR 104' 210
MEMORY 104 220
BB 103-105
SIGN/MAG
ACQCLK
PWRUP
SPI_CLK SK
SPI_DI SD
SPI_DO SI
SPI_CEB SSQ_N
RFRSTST SRESET_N
GPS OSCILLATOR 104-204
RF 103-103
103 107

FIG. 103

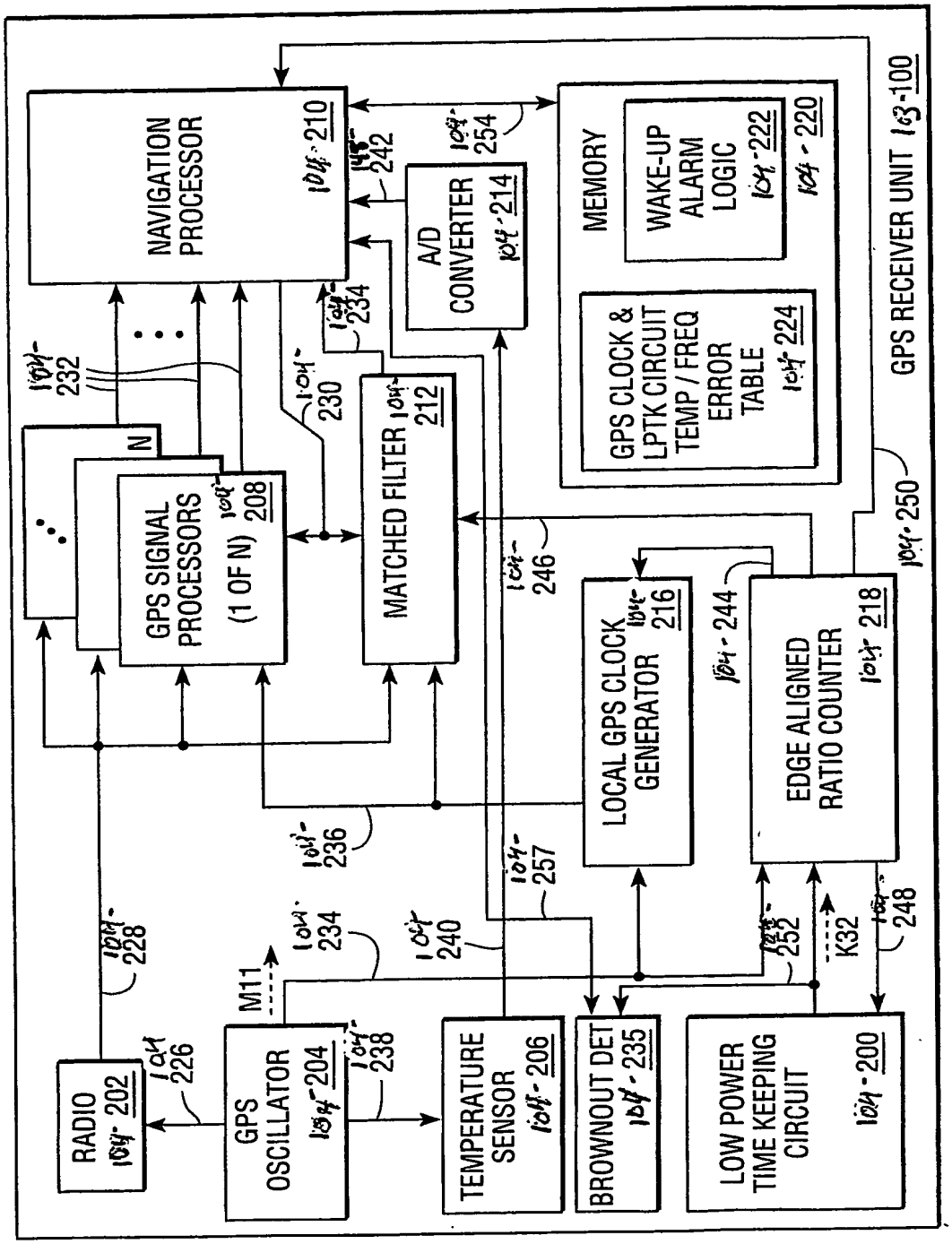
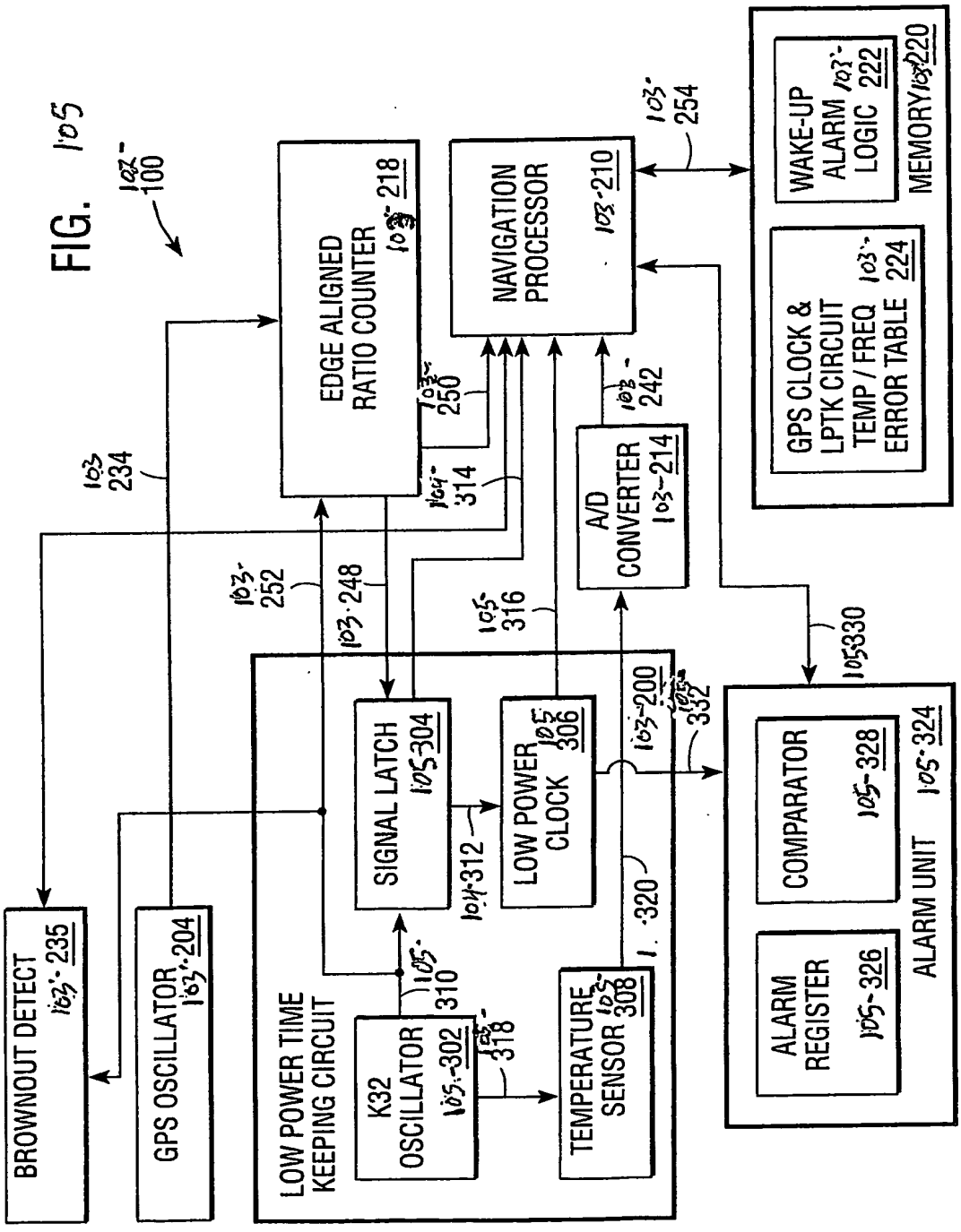

GPS RECEIVER UNIT 103-100
NAVIGATION PROCESSOR 210
A/D CONVERTER 214
MEMORY 220
WAKE-UP ALARM LOGIC 222
GPS CLOCK & LPTK CIRCUIT TEMP / FREQ ERROR TABLE 224
GPS SIGNAL PROCESSORS (1 OF N) 208
N
MATCHED FILTER 212
LOCAL GPS CLOCK GENERATOR 216
EDGE ALIGNED RATIO COUNTER 218
RADIO 202
GPS OSCILLATOR 204
TEMPERATURE SENSOR 206
BROWNOUT DET 235
LOW POWER TIME KEEPING CIRCUIT 200
M11
K32

FIG. 104

FIG. 105


102-100
EDGE ALIGNED RATIO COUNTER 218
NAVIGATION PROCESSOR 210
WAKE-UP ALARM LOGIC 222
MEMORY 220
GPS CLOCK & LPTK CIRCUIT TEMP / FREQ ERROR TABLE 224
A/D CONVERTER 214
BROWNOUT DETECT 235
GPS OSCILLATOR 204
LOW POWER TIME KEEPING CIRCUIT
SIGNAL LATCH 304
LOW POWER CLOCK 306
K32 OSCILLATOR 302
TEMPERATURE SENSOR 308
COMPARATOR 328
ALARM REGISTER 326
ALARM UNIT 324
234
252
248
250
314
242
254
316
312
310
318
320
200
332
330

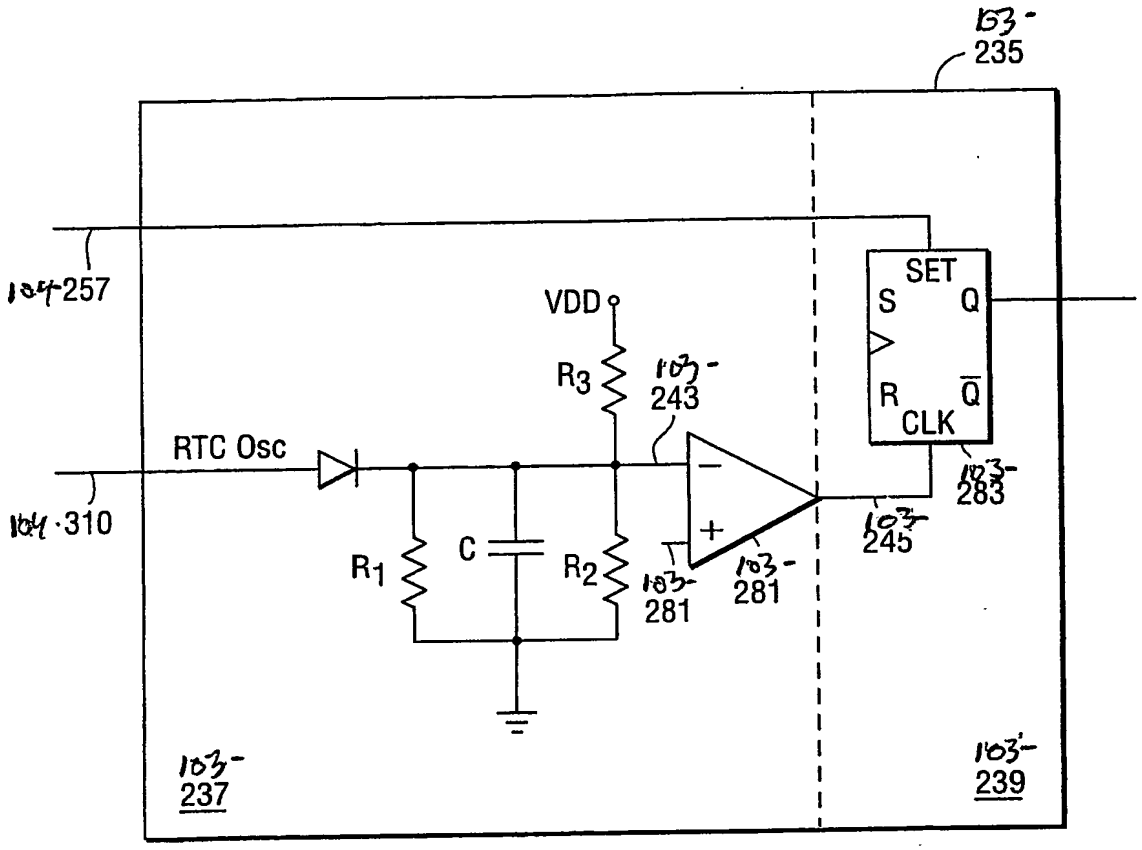

103-
235
104-257
VDD
R3
103-
243
SET
S
Q
R
Q
CLK
RTC Osc
104-310
R1
C
R2
103-
281
103-
281
103-
283
103-
245
103-
237
103-
239

FIG. 106

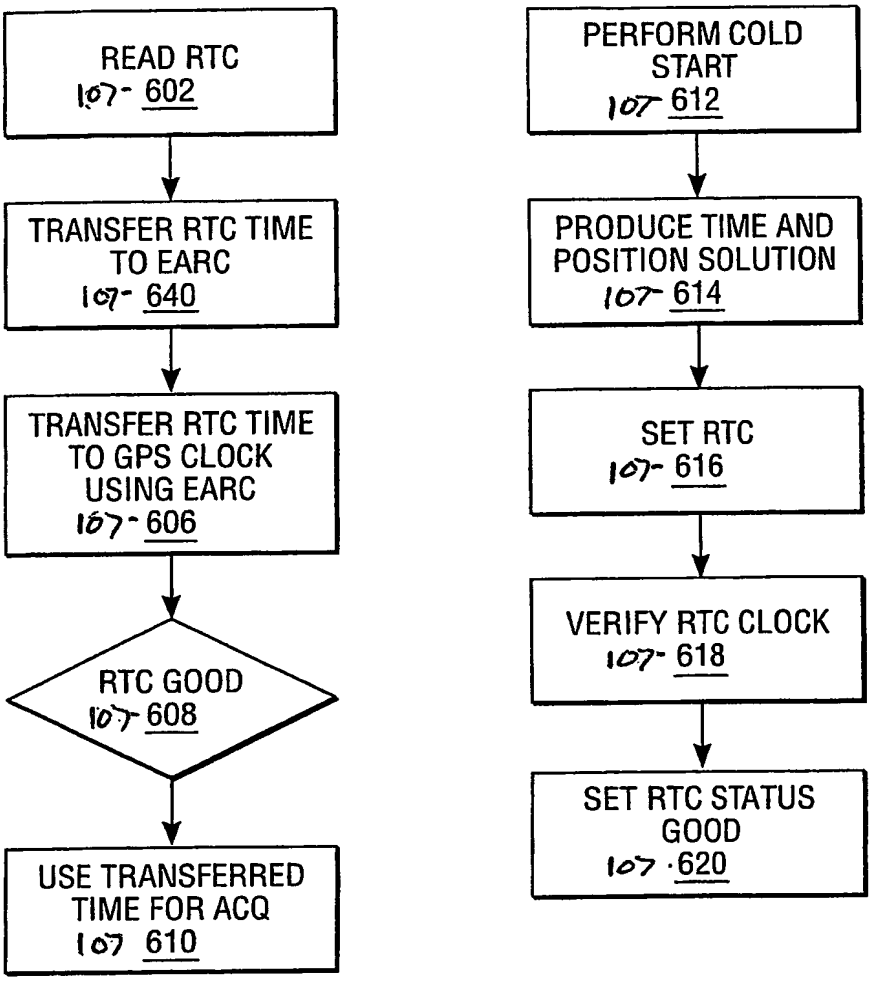

READ RTC
107- 602
TRANSFER RTC TIME TO EARC
107- 640
TRANSFER RTC TIME TO GPS CLOCK USING EARC
107- 606
RTC GOOD
107 608
USE TRANSFERRED TIME FOR ACQ
107 610
PERFORM COLD START
107 612
PRODUCE TIME AND POSITION SOLUTION
107- 614
SET RTC
107- 616
VERIFY RTC CLOCK
107- 618
SET RTC STATUS GOOD
107 ·620

FIG. 107

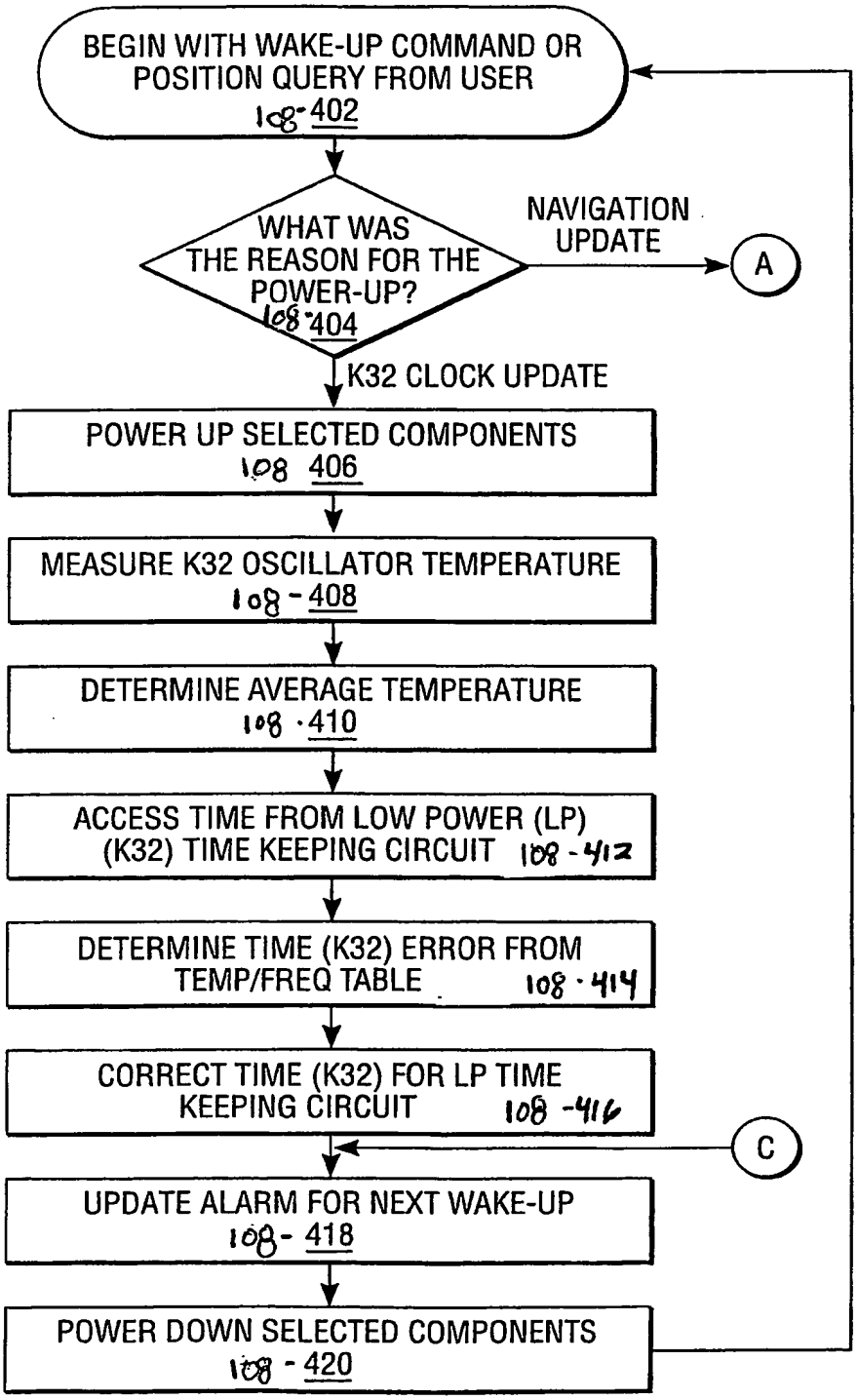


FIG. 108A

108-400

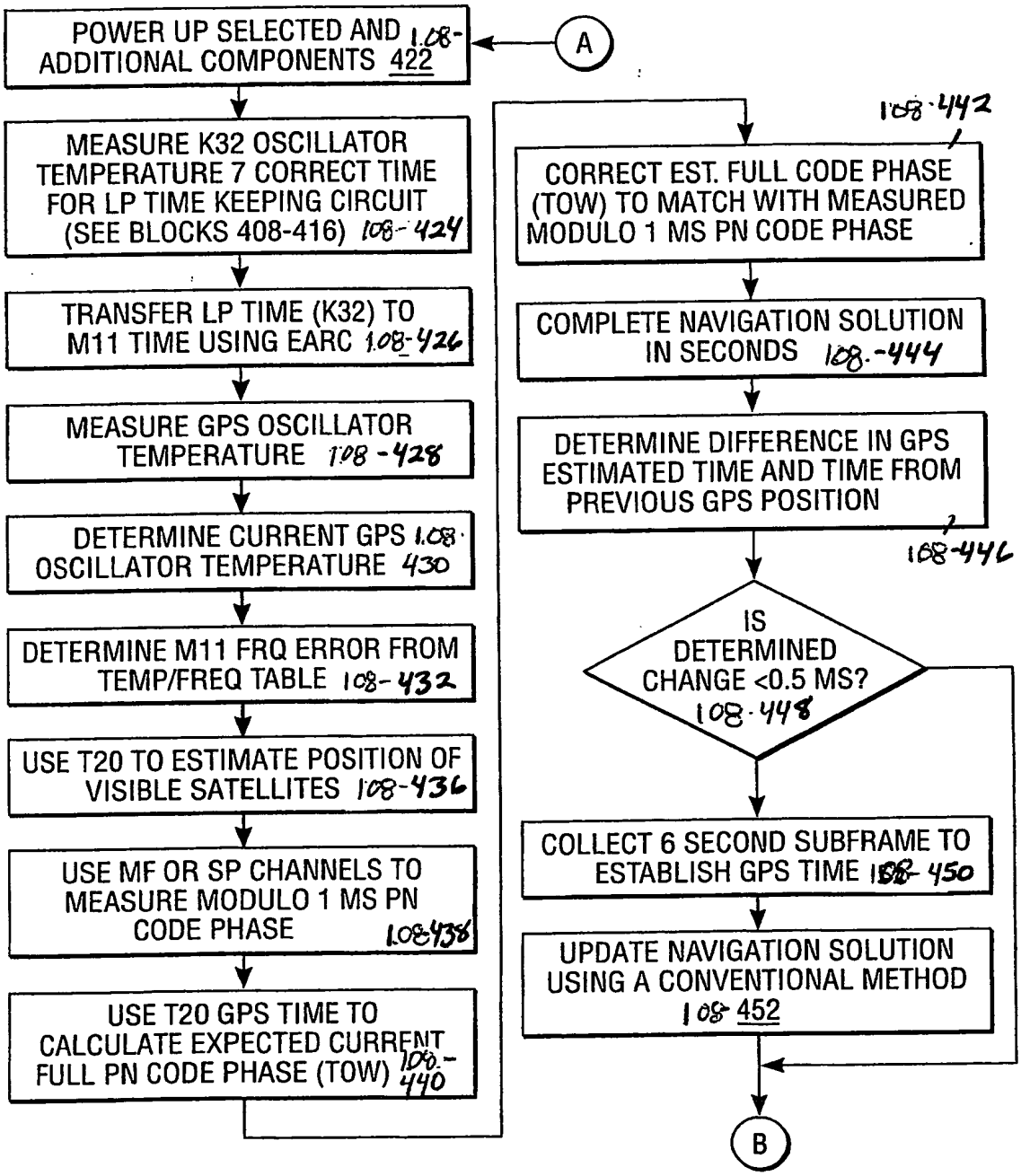

POWER UP SELECTED AND ADDITIONAL COMPONENTS 422
108-
A
MEASURE K32 OSCILLATOR TEMPERATURE 7 CORRECT TIME FOR LP TIME KEEPING CIRCUIT (SEE BLOCKS 408-416) 108-424
TRANSFER LP TIME (K32) TO M11 TIME USING EARC 108-426
MEASURE GPS OSCILLATOR TEMPERATURE 108-428
DETERMINE CURRENT GPS OSCILLATOR TEMPERATURE 430
DETERMINE M11 FRQ ERROR FROM TEMP/FREQ TABLE 108-432
USE T20 TO ESTIMATE POSITION OF VISIBLE SATELLITES 108-436
USE MF OR SP CHANNELS TO MEASURE MODULO 1 MS PN CODE PHASE 108-438
USE T20 GPS TIME TO CALCULATE EXPECTED CURRENT FULL PN CODE PHASE (TOW) 108-440
108-442
CORRECT EST. FULL CODE PHASE (TOW) TO MATCH WITH MEASURED MODULO 1 MS PN CODE PHASE
COMPLETE NAVIGATION SOLUTION IN SECONDS 108-444
DETERMINE DIFFERENCE IN GPS ESTIMATED TIME AND TIME FROM PREVIOUS GPS POSITION
108-446
IS DETERMINED CHANGE <0.5 MS?
108-448
COLLECT 6 SECOND SUBFRAME TO ESTABLISH GPS TIME 108-450
UPDATE NAVIGATION SOLUTION USING A CONVENTIONAL METHOD 108-452
B

FIG. 108B

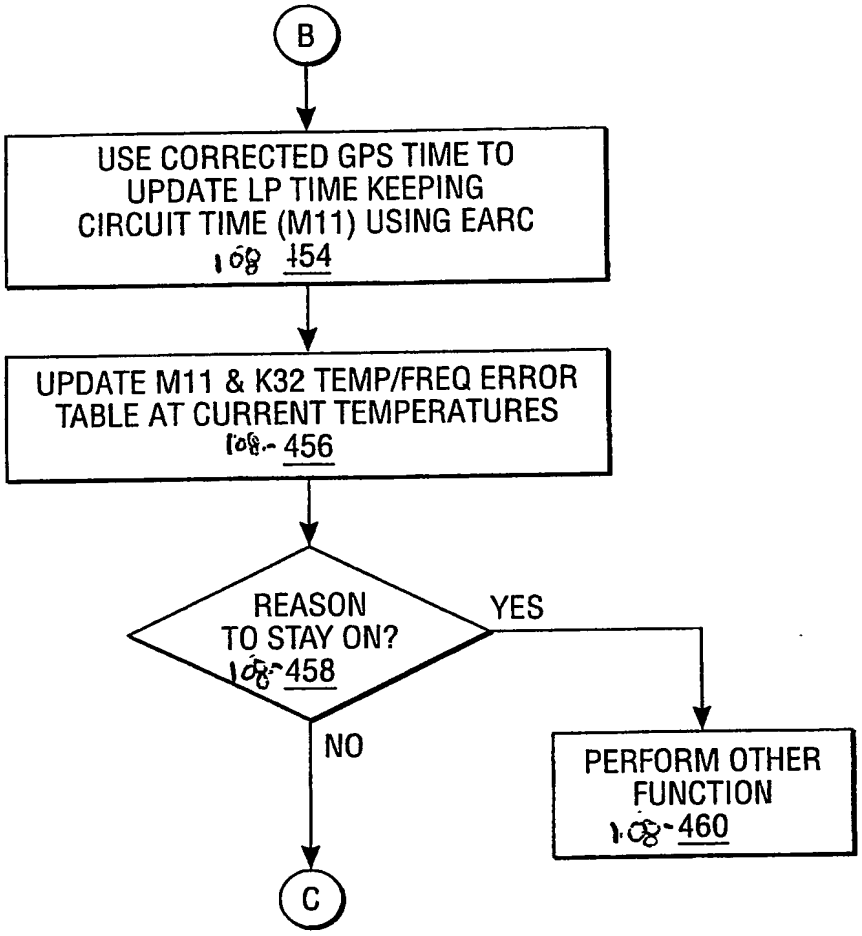

B
USE CORRECTED GPS TIME TO UPDATE LP TIME KEEPING CIRCUIT TIME (M11) USING EARC
108 154
UPDATE M11 & K32 TEMP/FREQ ERROR TABLE AT CURRENT TEMPERATURES
108-456
REASON TO STAY ON?
108-458
YES
NO
PERFORM OTHER FUNCTION
108-460
C

FIG. 108C

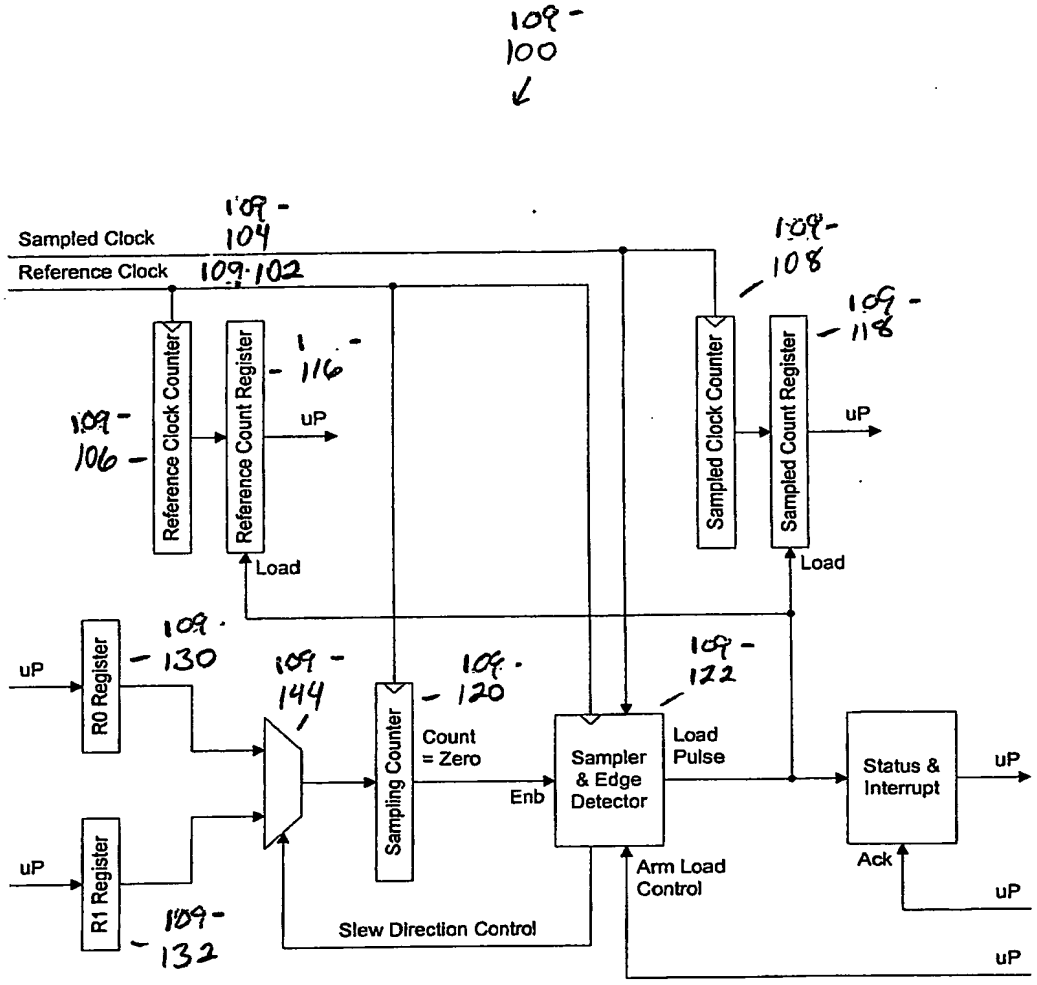

109-100
109-104
Sampled Clock
Reference Clock
109-102
109-108
109-118
109-116
109-106
Reference Clock Counter
Reference Count Register
uP
Sampled Clock Counter
Sampled Count Register
uP
Load
Load
109-130
109-144
109-120
109-122
uP
R0 Register
Sampling Counter
Count = Zero
Sampler & Edge Detector
Load Pulse
Status & Interrupt
uP
uP
R1 Register
Enb
Arm Load Control
Ack
uP
109-132
Slew Direction Control
uP

FIGURE 109

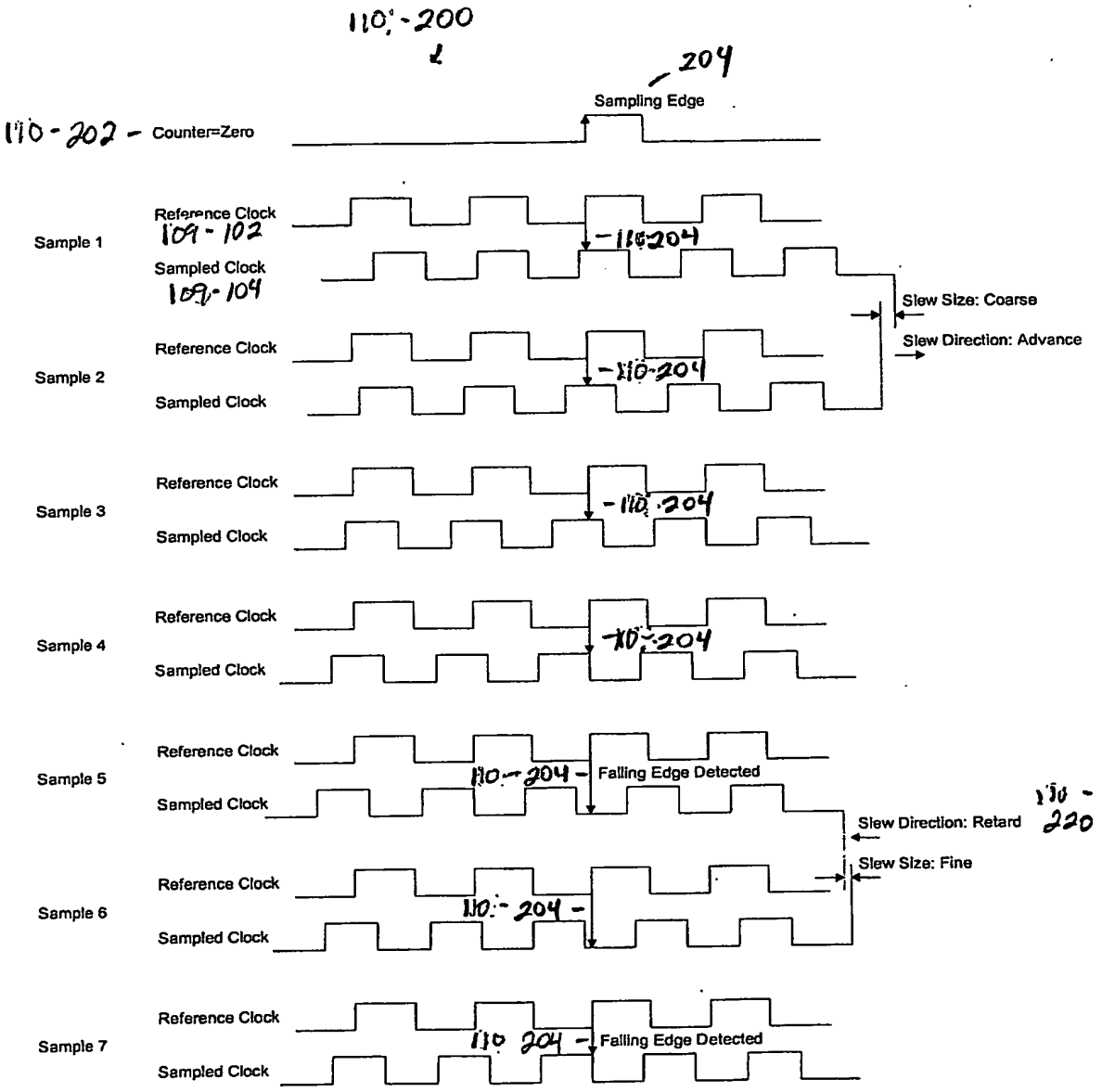

110-200
204
Sampling Edge
110-202 - Counter=Zero
Sample 1
Reference Clock
109-102
110-204
Sampled Clock
109-104
Slew Size: Coarse
Slew Direction: Advance
Sample 2
Reference Clock
110-204
Sampled Clock
Sample 3
Reference Clock
110-204
Sampled Clock
Sample 4
Reference Clock
110-204
Sampled Clock
Sample 5
Reference Clock
110-204 - Falling Edge Detected
Sampled Clock
110-220
Slew Direction: Retard
Slew Size: Fine
Sample 6
Reference Clock
110-204 -
Sampled Clock
Sample 7
Reference Clock
110 204 - Falling Edge Detected
Sampled Clock

FIGURE 110

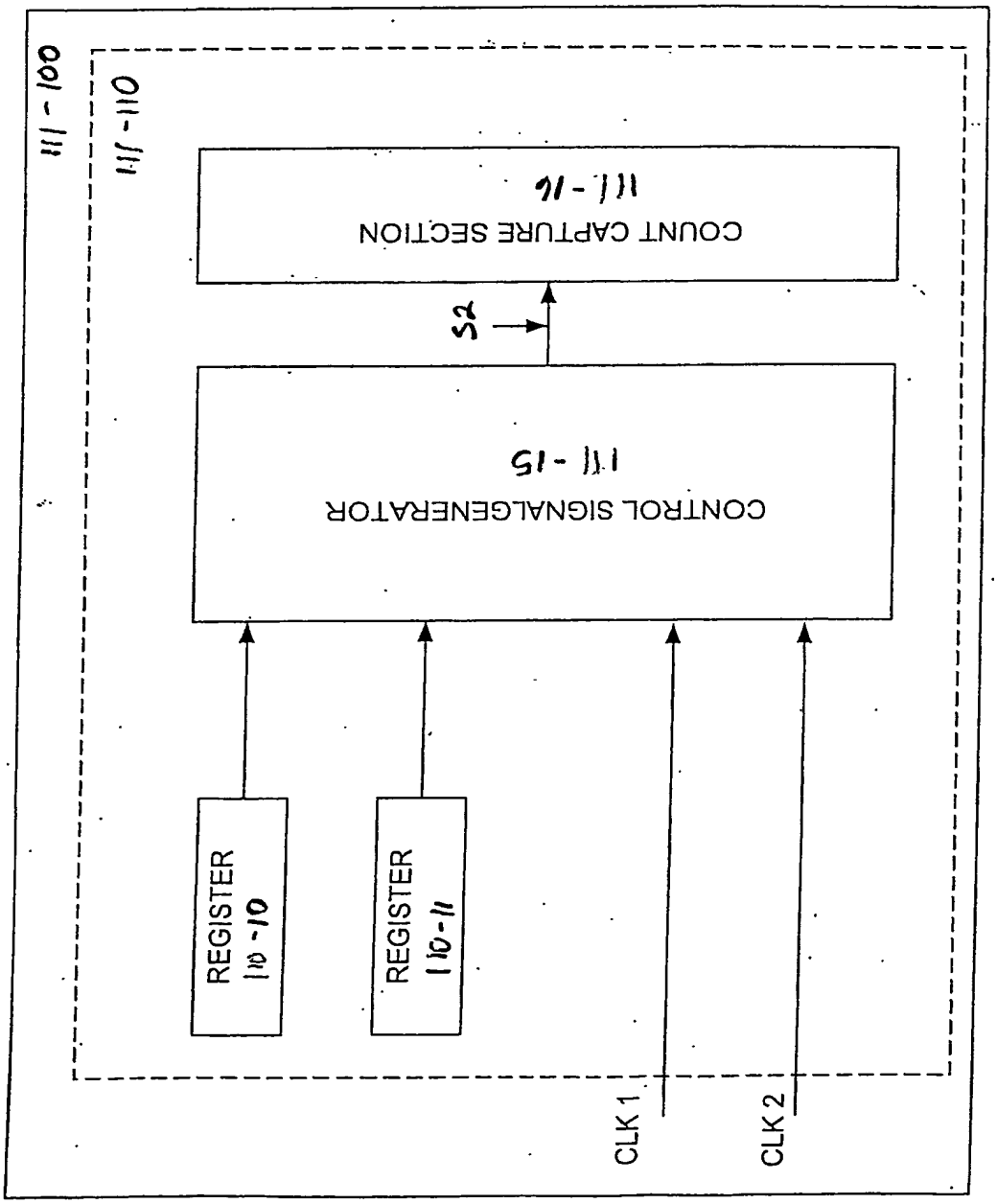

111 - 100
111 - 110
COUNT CAPTURE SECTION
111 - 16
SA
CONTROL SIGNALGENERATOR
111 - 15
REGISTER
110 - 10
REGISTER
110 - 11
CLK 1
CLK 2

FIG. 111

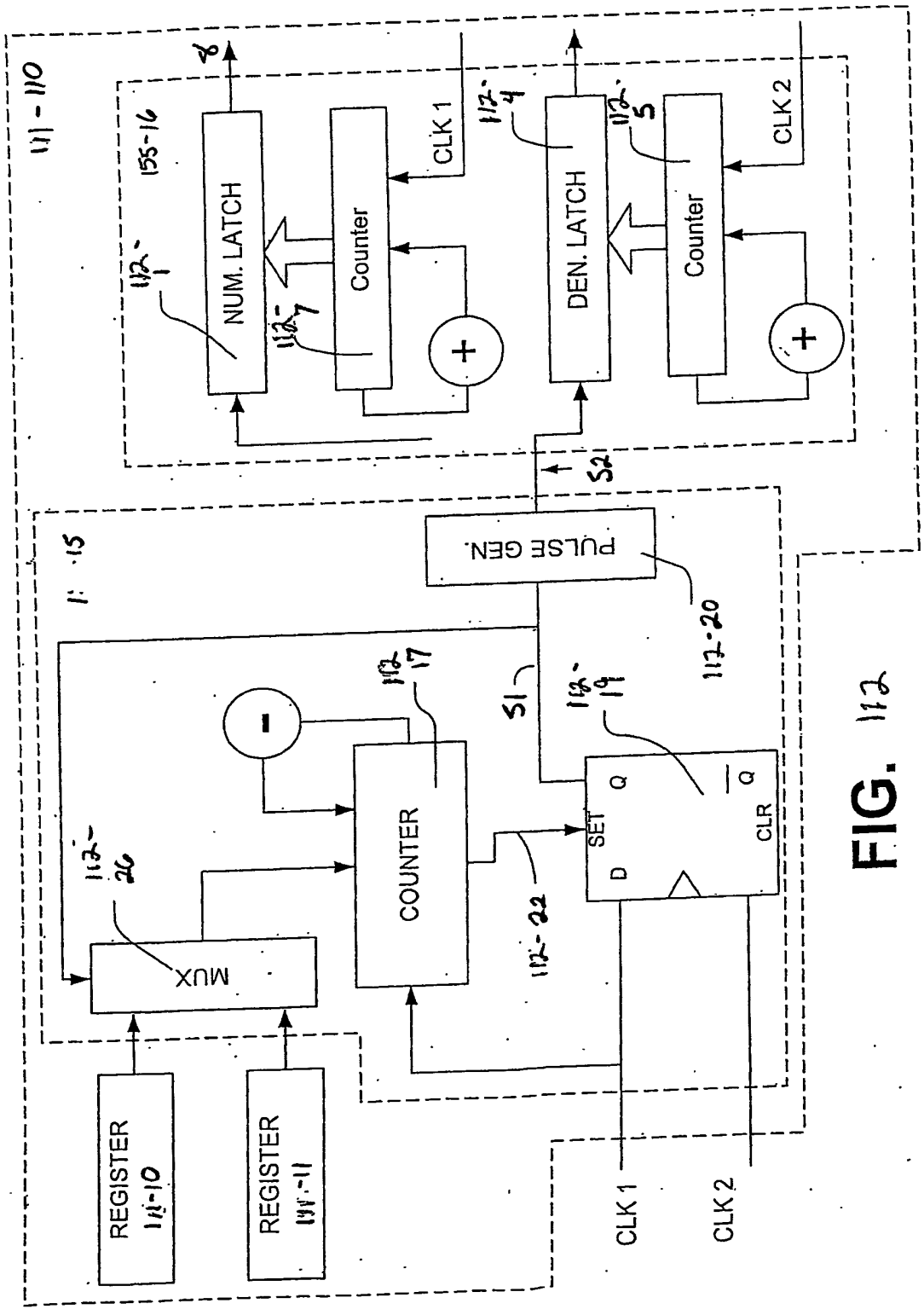

111 - 110
115-16
112-1
NUM. LATCH
112-7
Counter
CLK 1
112-4
DEN. LATCH
112-5
Counter
CLK 2
+
+
52
PULSE GEN.
112-20
51
112-19
112-17
-
COUNTER
112-22
112-26
MUX
SET
D
Q
Q
CLR
REGISTER 111-10
REGISTER 111-11
CLK 1
CLK 2

FIG. 112

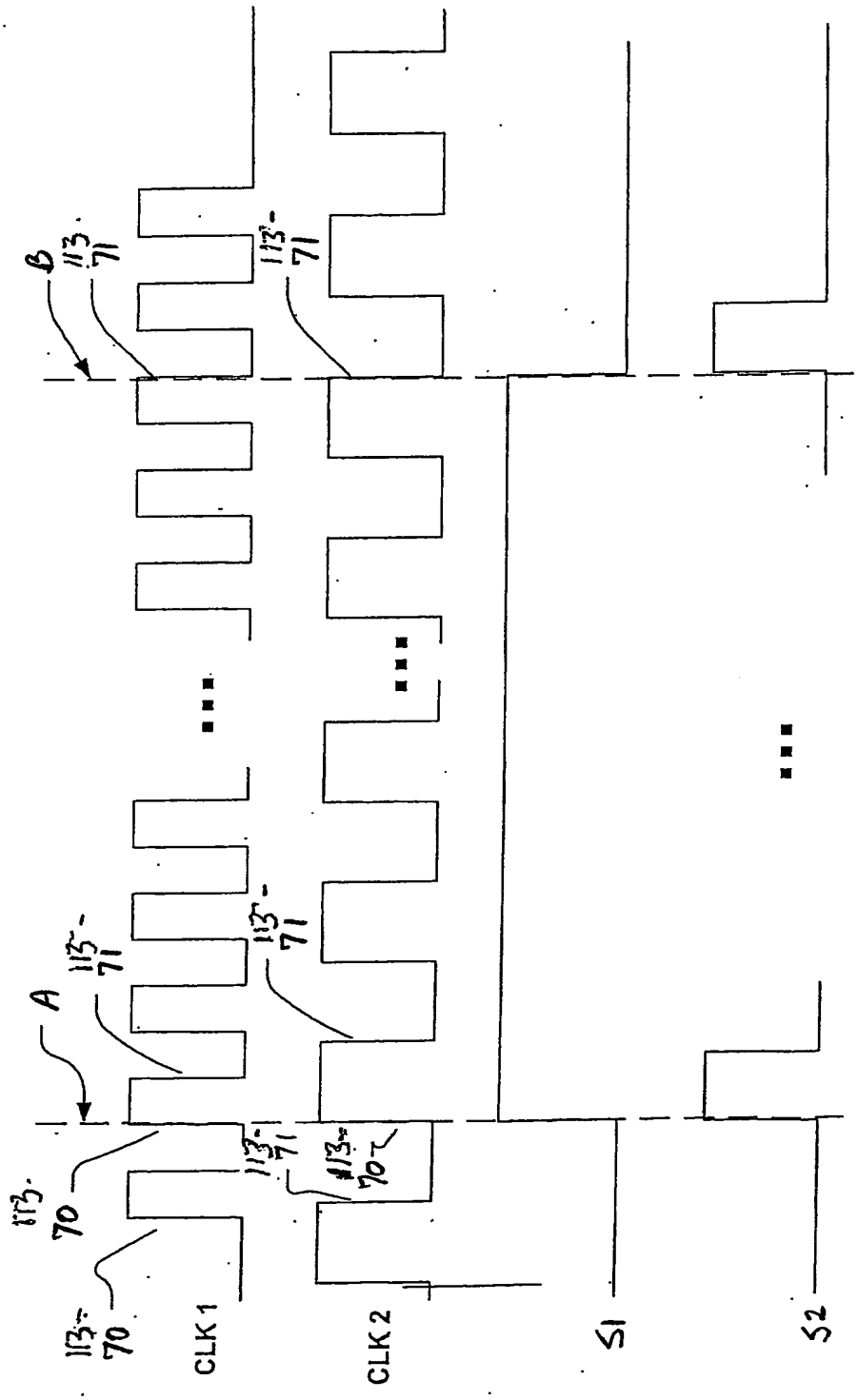

B
113-71
113-71
A
113-71
113-71
113-70
113-70
113-71
113-70
CLK 1
CLK 2
S1
S2

FIG. 113

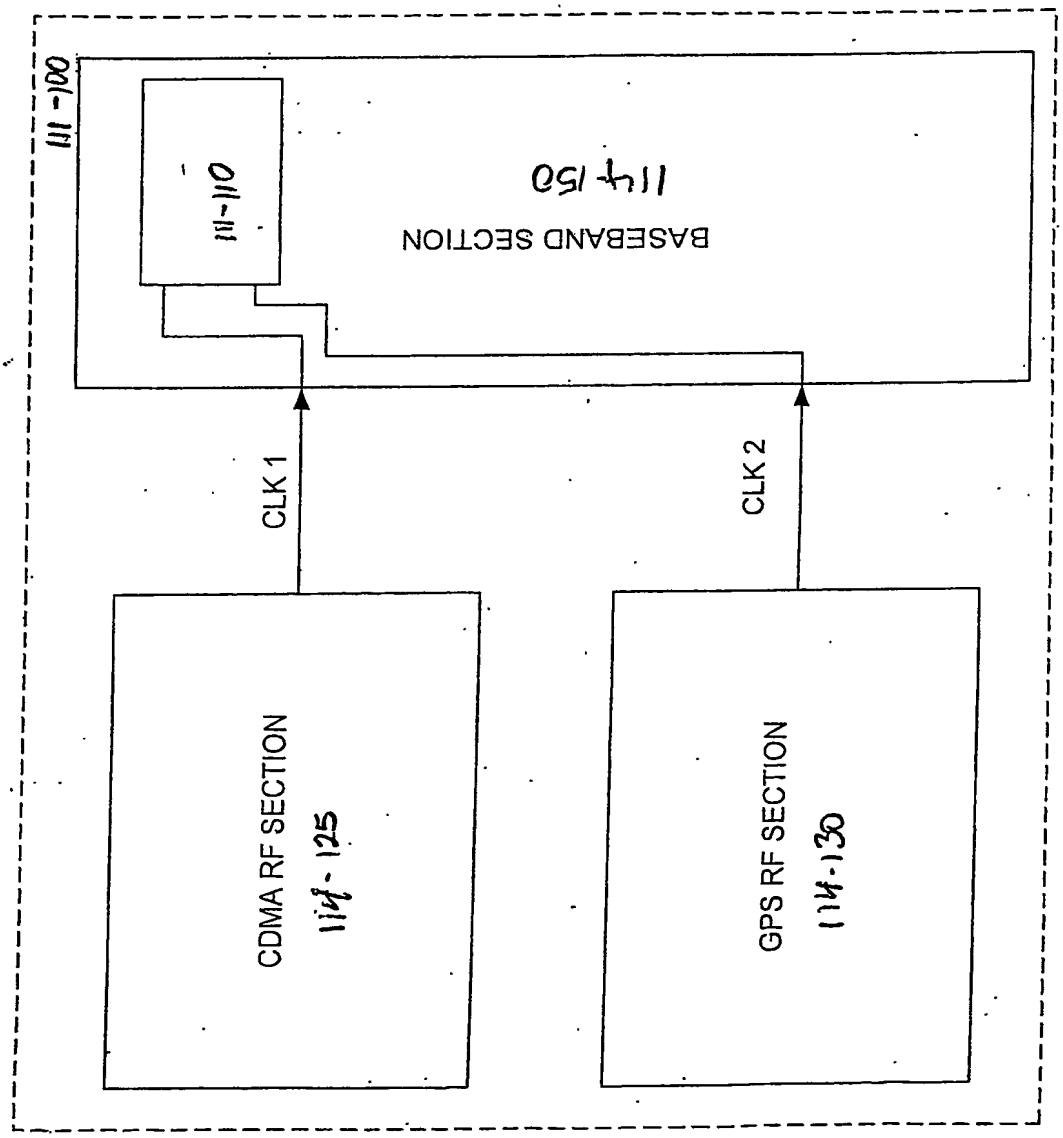

111-100
111-110
BASEBAND SECTION
114-150
CLK 1
CLK 2
CDMA RF SECTION
114-125
GPS RF SECTION
114-130

FIG. 114

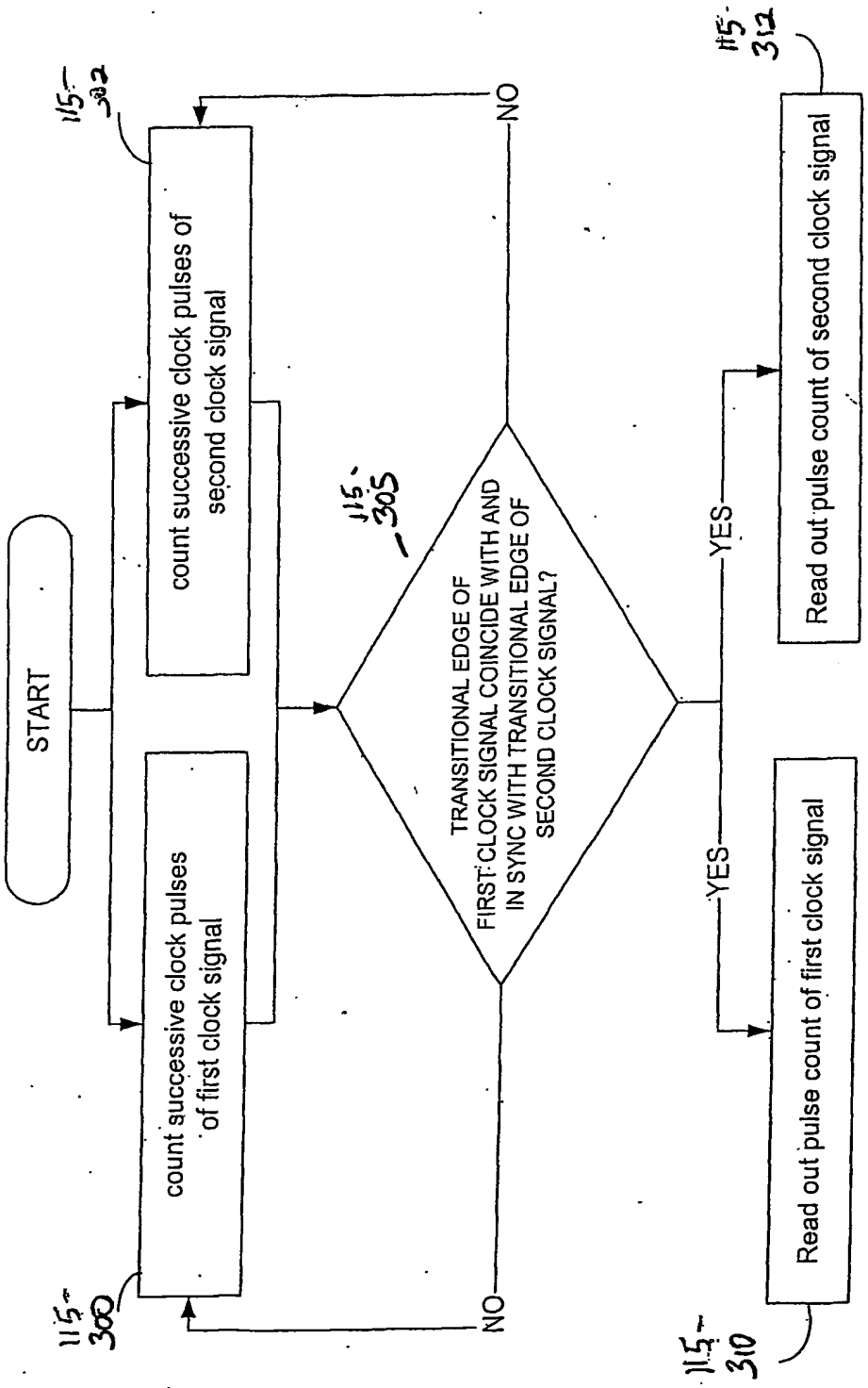

START
count successive clock pulses of first clock signal
count successive clock pulses of second clock signal
TRANSITIONAL EDGE OF FIRST CLOCK SIGNAL COINCIDE WITH AND IN SYNC WITH TRANSITIONAL EDGE OF SECOND CLOCK SIGNAL?
YES
NO
YES
NO
Read out pulse count of first clock signal
Read out pulse count of second clock signal
115-300
115-302
115-305
115-310
115-312

FIG. 115

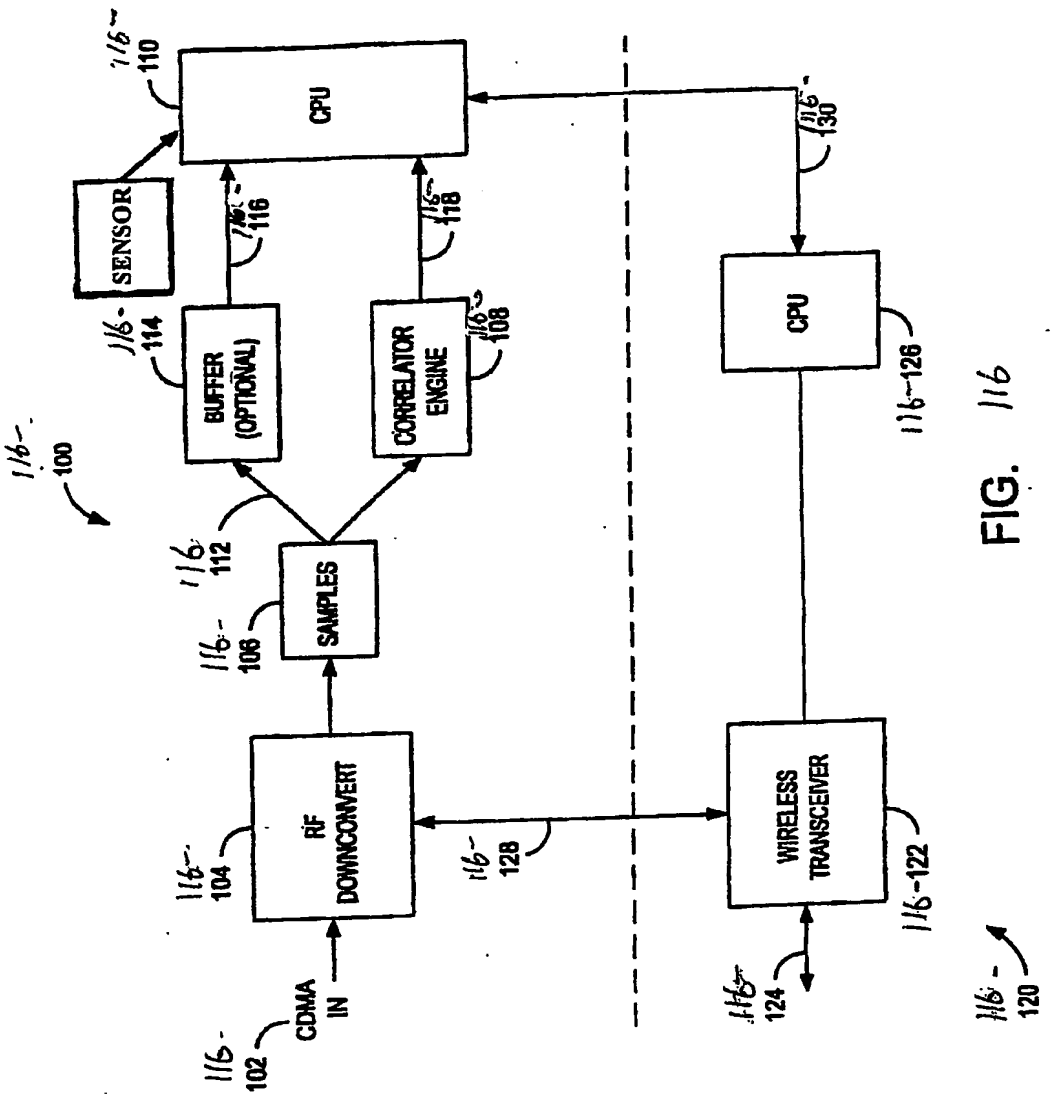

116
100
CDMA IN
116-102
RF DOWNCONVERT
116-104
SAMPLES
116-106
CORRELATOR ENGINE
116-108
BUFFER (OPTIONAL)
116-114
SENSOR
CPU
116-110
116-112
116-116
116-118
116-128
WIRELESS TRANSCEIVER
116-122
116-124
CPU
116-126
116-130
116-120

FIG. 116

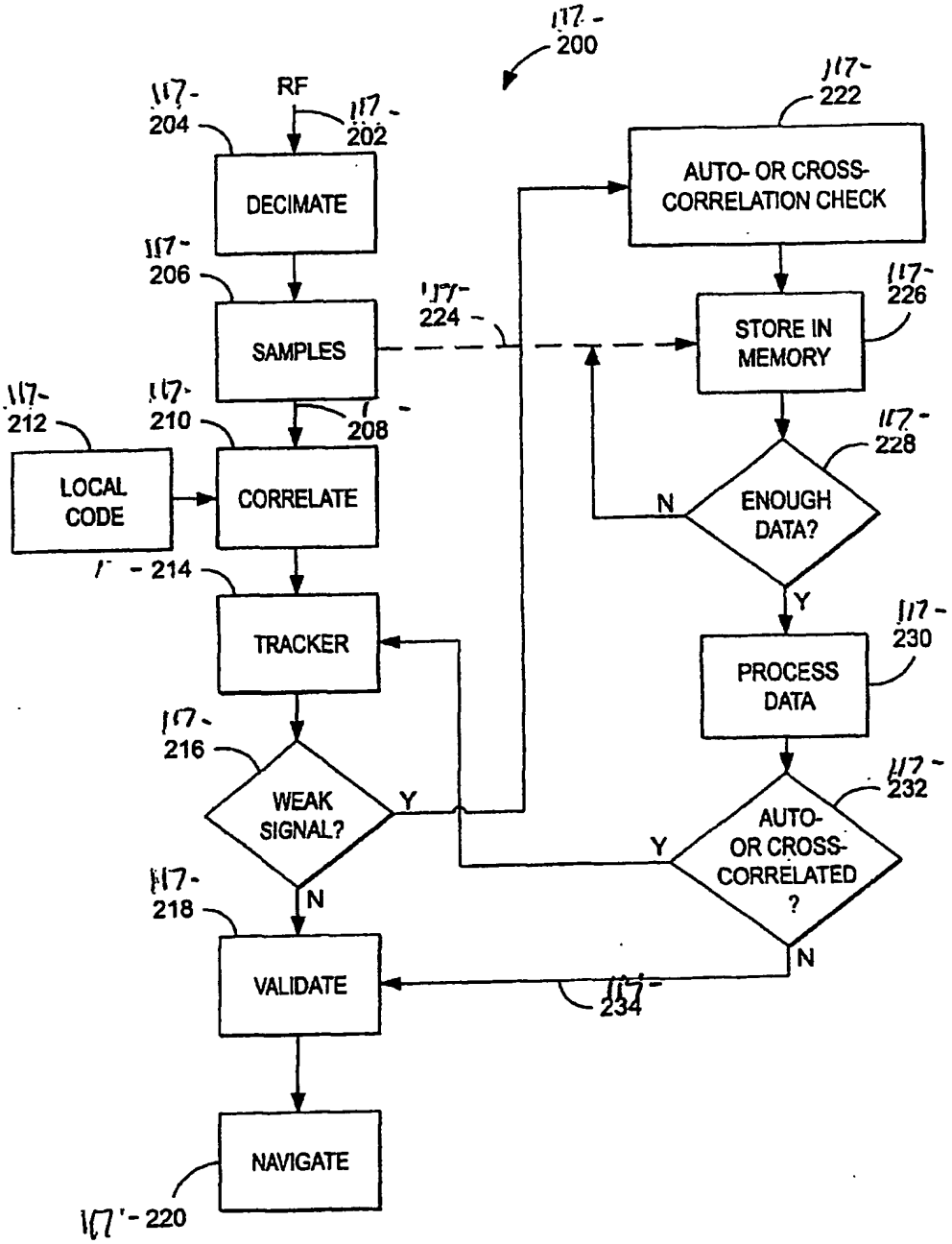

117-200
RF
117-202
117-204
DECIMATE
117-206
SAMPLES
208
117-210
CORRELATE
117-212
LOCAL CODE
214
TRACKER
117-216
WEAK SIGNAL?
Y
N
117-218
VALIDATE
117-220
NAVIGATE
117-222
AUTO- OR CROSS-CORRELATION CHECK
117-224
117-226
STORE IN MEMORY
117-228
ENOUGH DATA?
N
Y
117-230
PROCESS DATA
117-232
AUTO- OR CROSS-CORRELATED ?
Y
N
117-234

FIG. 117

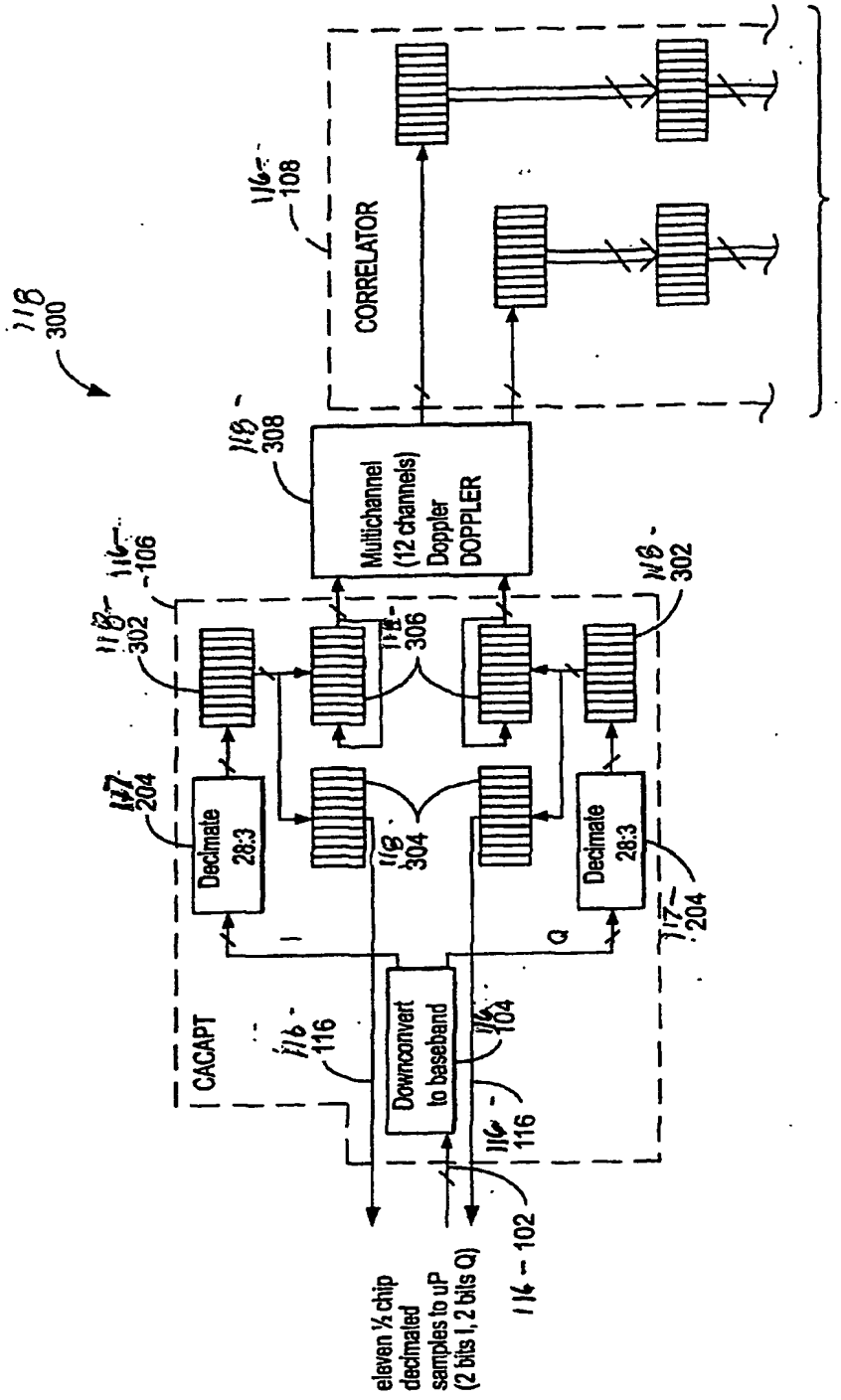

118
300
CORRELATOR
116-108
To Fig. 3B
118-308
Multichannel (12 channels) Doppler DOPPLER
116-106
118-302
118-306
118-304
117-204
Decimate 28:3
I
Q
CACAPT
116-116
Downconvert to baseband
116-104
116-116
116-102
eleven ½ chip decimated samples to uP (2 bits I, 2 bits Q)

FIG. 118-A

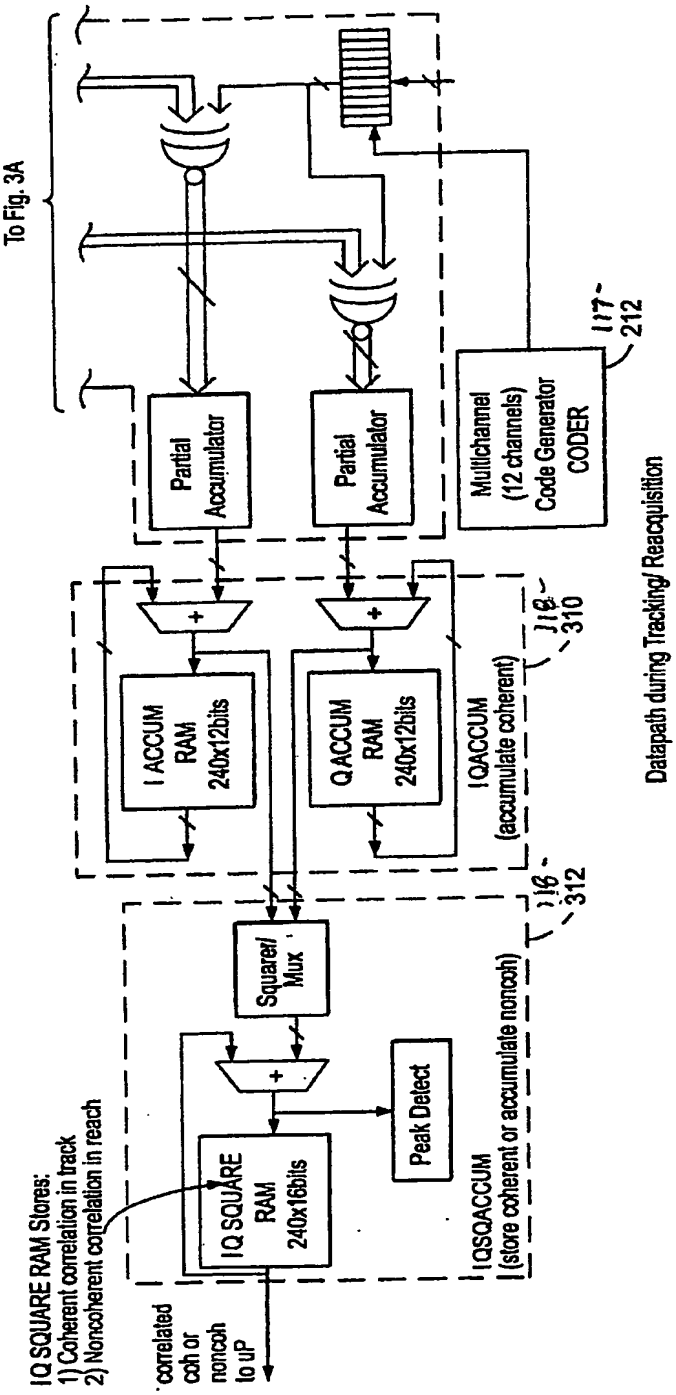

To Fig. 3A
Partial Accumulator
Partial Accumulator
Multichannel (12 channels) Code Generator CODER
117-212
Datapath during Tracking/ Reacquisition
I ACCUM RAM 240x12bits
Q ACCUM RAM 240x12bits
IQACCUM (accumulate coherent)
118-310
118-312
Squarer/ Mux
Peak Detect
IQ SQUARE RAM 240x16bits
IQSQACCUM (store coherent or accumulate noncoh)
IQ SQUARE RAM Stores:
1) Coherent correlation in track
2) Noncoherent correlation in reach
correlated coh or noncoh to uP

FIG. 18B

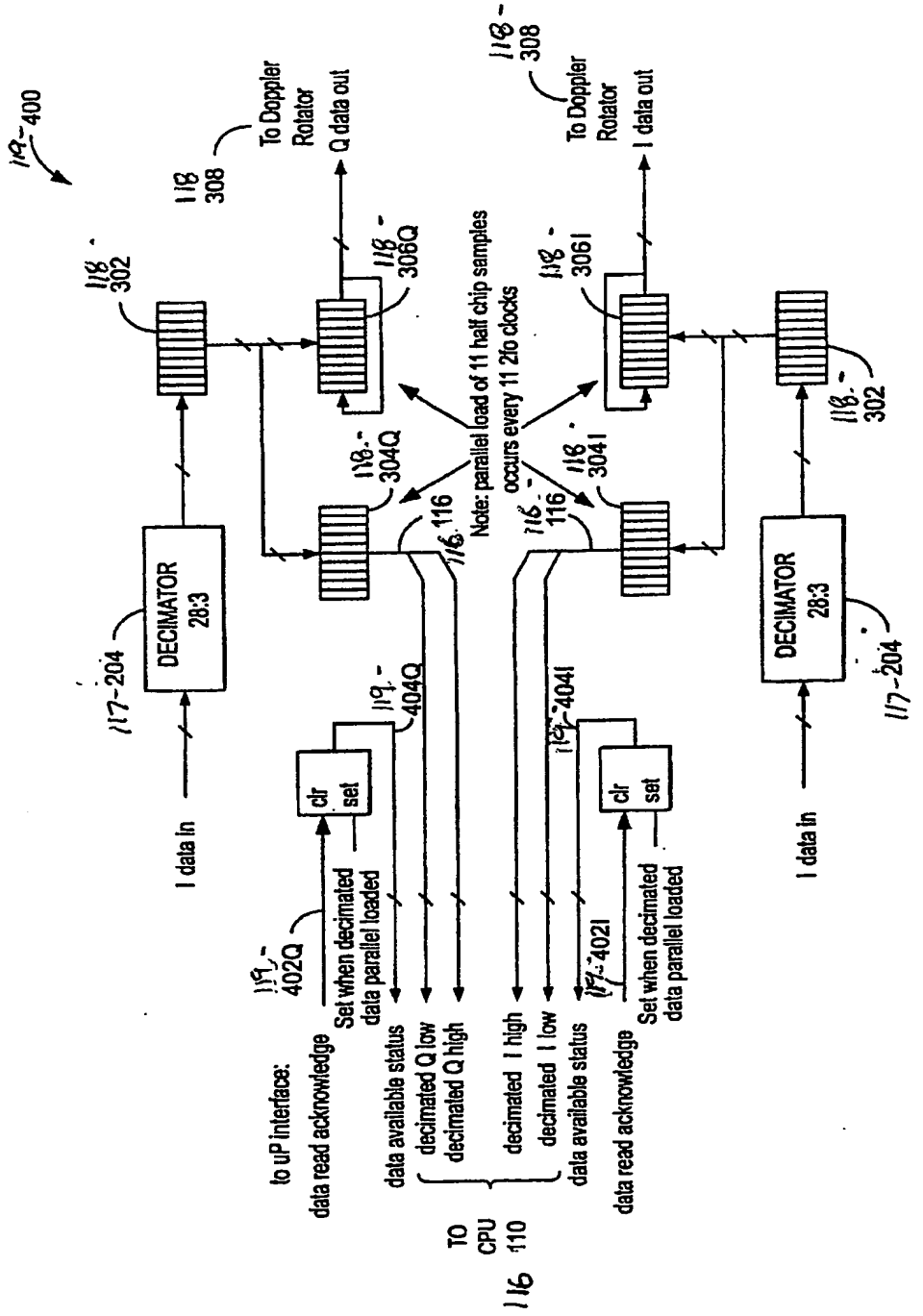

400
118 308
To Doppler Rotator
Q data out
118 - 308
To Doppler Rotator
I data out
118 - 302
118 - 306Q
118 - 306I
118 - 302
Note: parallel load of 11 half chip samples occurs every 11 2fo clocks
118 - 304Q
118 - 304I
116
116
116
116
DECIMATOR 28:3
DECIMATOR 28:3
117 - 204
117 - 204
I data in
I data in
clr
set
clr
set
119 - 404Q
119 - 404I
119 - 402Q
119 - 402I
Set when decimated data parallel loaded
Set when decimated data parallel loaded
to uP interface:
data read acknowledge
data available status
decimated Q low
decimated Q high
decimated I high
decimated I low
data available status
data read acknowledge
TO CPU 110
116
CA CAPTURE BLOCK

FIG. 19

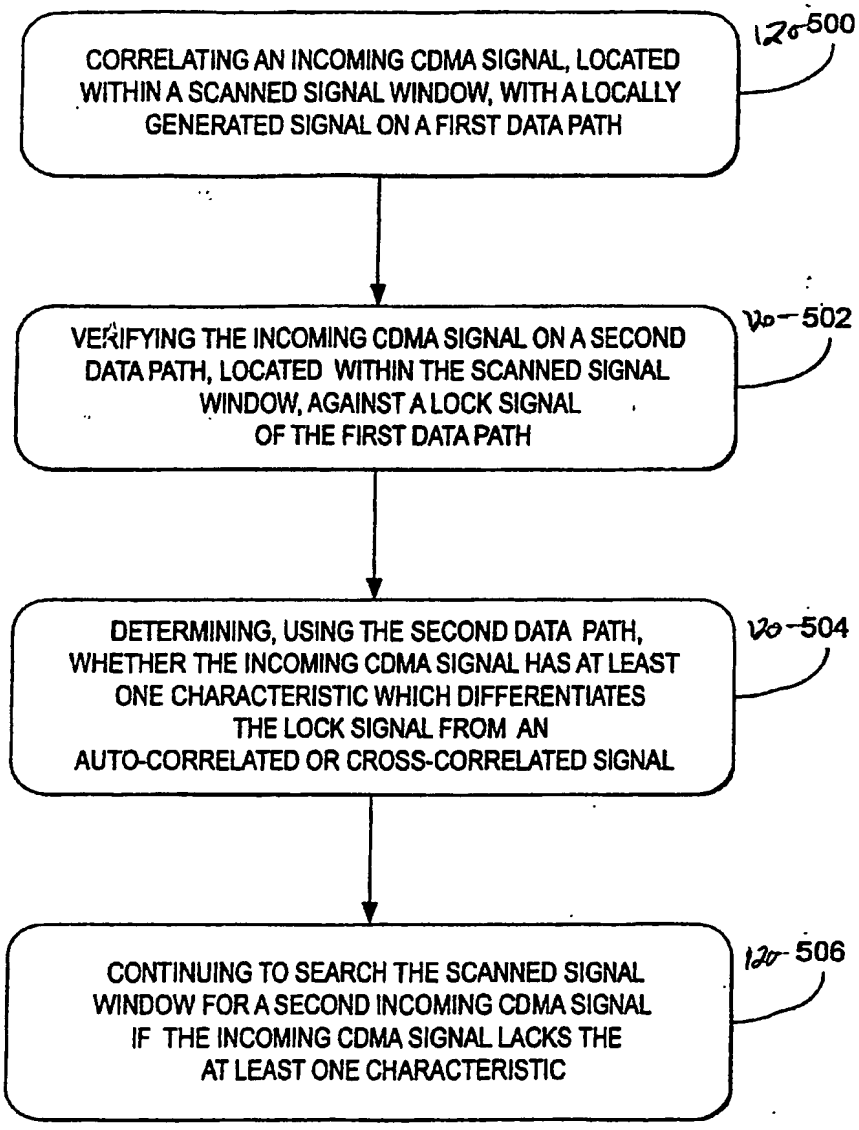

CORRELATING AN INCOMING CDMA SIGNAL, LOCATED WITHIN A SCANNED SIGNAL WINDOW, WITH A LOCALLY GENERATED SIGNAL ON A FIRST DATA PATH
120-500
VERIFYING THE INCOMING CDMA SIGNAL ON A SECOND DATA PATH, LOCATED WITHIN THE SCANNED SIGNAL WINDOW, AGAINST A LOCK SIGNAL OF THE FIRST DATA PATH
120-502
DETERMINING, USING THE SECOND DATA PATH, WHETHER THE INCOMING CDMA SIGNAL HAS AT LEAST ONE CHARACTERISTIC WHICH DIFFERENTIATES THE LOCK SIGNAL FROM AN AUTO-CORRELATED OR CROSS-CORRELATED SIGNAL
120-504
CONTINUING TO SEARCH THE SCANNED SIGNAL WINDOW FOR A SECOND INCOMING CDMA SIGNAL IF THE INCOMING CDMA SIGNAL LACKS THE AT LEAST ONE CHARACTERISTIC
120-506

FIG. 120